NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

A STUDY OF TOPIC AND TOPIC CHANGE IN CONVERSATIONAL THREADS

by

Jessy Cowan-Sharp

September 2009

Thesis Advisor:	Craig H. Martell
Second Reader:	Simson L. Garfinkel

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
25–9–2009	Master's Thesis	2009-01-01—2009-09-25

4. TITLE AND SUBTITLE

A Study of Topic and Topic Change in Conversational Threads

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Jessy Cowan-Sharp

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Department of the Navy and NASA Ames Research Center

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This thesis applies Latent Dirichlet Allocation (LDA) to the problem of topic and topic change in conversational threads using e-mail. We demonstrate that LDA can be used to successfully classify raw e-mail messages with threads to which they belong, and compare the results with those for processed threads, where quoted and reply text have been removed. Raw thread classification performs better, but processed threads show promise. We then present two new, unsupervised techniques for identifying topic change in e-mail. The first is a keyword clustering approach using LDA and DBSCAN to identify clusters of topics, and transition points between them. The second is a sliding window technique which assesses the current topic for every window, identifying transition points. The keyword clustering performs better than the sliding window approach. Both can be used as a baseline for future work.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER *(include area code)*
Unclassified	Unclassified	Unclassified	UU	95	

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

A STUDY OF TOPIC AND TOPIC CHANGE IN CONVERSATIONAL THREADS

Jessy Cowan-Sharp
Civilian, NASA Ames Research Center
B.S., Queen's University, 2003

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2009

Author: Jessy Cowan-Sharp

Approved by: Craig H. Martell
Thesis Advisor

Simson L. Garfinkel
Second Reader

Dr. Peter Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis applies Latent Dirichlet Allocation (LDA) to the problem of topic and topic change in conversational threads using e-mail. We demonstrate that LDA can be used to successfully classify raw e-mail messages with threads to which they belong, and compare the results with those for processed threads, where quoted and reply text have been removed. Raw thread classification performs better, but processed threads show promise. We then present two new, unsupervised techniques for identifying topic change in e-mail. The first is a keyword clustering approach using LDA and DBSCAN to identify clusters of topics, and transition points between them. The second is a sliding window technique which assesses the current topic for every window, identifying transition points. The keyword clustering performs better than the sliding window approach. Both can be used as a baseline for future work.

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

1 Introduction **1**
1.1 Introduction 1
1.2 Motivation 1
1.3 Applications 3
1.4 Relationship to Space Exploration 4
1.5 Outline of this Thesis. 5

2 Prior and Related Work **7**
2.1 Topic Detection 7
2.2 Topic Change. 8
2.3 Conversation Flow. 10
2.4 E-Mail 10

3 Technical Concepts and Data Processing **13**
3.1 Technical Concepts 13
3.2 Data Processing 23

4 Experiments **29**
4.1 Message Thread Classification using LDA 29
4.2 Automated Detection of Topic Change 36

5 Analysis and Future Work **47**
5.1 E-Mail Thread Classification. 47
5.2 Topic Change. 51
5.3 Stopwords and Keywords 58
5.4 Signature Detection 60
5.5 Other Future Work. 62

6 Conclusions **63**

List of References **65**

A Code **67**
A.1 DBSCAN 67

B Stop Words **71**
B.1 Generic Stop Words 71
B.2 Custom Stop Words 74

Referenced Authors **77**

List of Figures

3.1 Basic word tokenization and vector-space representation for unigrams. 14

3.2 Symmetric Dirichlet distribution for three topics on a 2-dimensional simplex. Darker colours indicate higher probability. Left: $\alpha = 4$; right: $\alpha = 2$. Figure and caption from [19, p.5]. 17

3.3 Illustration of points that would be considered border points and core points in the DBSCAN algorithm. Figure from [7, p.3]. 21

3.4 DBSCAN density-reachable and density-connected points. Figure from [7, p.3] 22

3.5 High-level overview of steps taken for data pre-processing, and both experiments. 24

3.6 Overview of thread structure and terms, showing in-reply-to structure and longest branch. 26

4.1 E-Mail classification performance for LDA models built with different numbers of topics, for both raw and processed threads. Each line shows a different distance metric. It can be seen that varying the LDA topics parameter improves the results. 32

4.2 A Comparison of e-mail classification performance for LDA models built with increasing numbers of threads. Results are shown for both processed threads (lower group) and raw threads (upper group). Each line shows a different distance metric. Performance decreases as the number of threads goes up. 34

4.3 Performance as the number of LDA topics is increased, for 1000 processed threads. 35

4.4 This figure shows the keywords per topic sorted in order of decreasing weight (or relevance) to that topic. The x-axis shows the word ordering, while the y-axis shows the weight of a word for a given topic. There is one line per topic. The words themselves are not displayed; rather, the chart is shown to emphasize the exponential falloff of word-topic weights. 37

4.5 Topic correlations for keywords in a single e-mail thread. LDA Topics = 50; Keywords = 10; Threshold = 5. We can see that structure emerges in the thread. Note that the x-axis shows words in the order they appeared in the thread, and thus also correlates with time. 38

4.6 An example of the clustering results for a single thread. Threshold T=25, N=10 $min\_pts = 4$ and $\epsilon = 6$ (LDA Topics = 50). Vertical lines show identified topic transitions. Personally identifying terms are grayed out for privacy purposes. . 41

5.1 DBSCAN clusters are too far apart, resulting in meaningless transition. . . . 53

5.2 The end of the last cluster is a meaningful transition, but is not captured because there is no subsequent cluster. The transition identified *between* clusters of the same topic is an artifact of the experiment parameters, and not a real transition. 54

5.3 Three outputs from the sliding window experiment. Window sizes as described in respective titles. A) Correct results; B) Erroneously detects two topics, close to the actual topic change; C) Extreme overfitting, too many topics detected. LDA Topics = 50). Personally identifying terms are grayed out for privacy purposes. 56

5.4 In this graph of a subsection of a thread, we can see that stopwords consistently appear in two topics above and beyond all others. Personally identifying names and locations have been blurred. 59

5.5 A graph of topic word weight for one e-mail. Again the words along the x-axis are in order of appearance, so this dimension also represents time. 60

5.6 Power distribution of relationships in e-mail. Figure from [18, p.8] 61

List of Tables

4.1 Baseline accuracy values for different control groups. 29

4.2 LDA Parameters for LDA model of e-mail threads 30

4.3 Gibbs Parameters for LDA model of e-mail threads 30

4.4 Top ten words for 5 of the 50 topics in one of the LDA models, ordered by decreasing weight from the top. Note that these words are representative, and in practice change slightly for each run due to Gibbs sampling. This example also includes stop words which were subsequently removed. 30

4.5 Results from LDA classification experiment for each of Cosine, Euclidean, and Shannon and KL-divergence distance measures for processed threads. Since the results are a function of sampling, this is an average over 5 runs. Numbers may not sum to 100 due to rounding. 31

4.6 Results from LDA classification experiment for each of Cosine, Euclidean, and Shannon and KL-divergence distance measures *for raw threads*. Since the results are a function of sampling, this is an average over 5 runs. Numbers may not sum to 100 due to rounding. 31

4.7 High level effects of varying the parameters N and T in the selection of keywords used in topic identification. 38

4.8 DBSCAN results for (N=25, T=5, E=6, MP=4). Results show the Thread ID, thread length in characters (len (ch)), number of messages in thread branch, the number of actual topic changes in the thread (# Changes), the number of topic changes found (# Found), and the number found in the correct location (Cor. Loc). P, R, and F give the precision, recall and F-score, as defined in 3.1.7 . . 42

4.9 (DBSCAN results for N=25, T=5, E=10, MP=6). Results show the Thread ID, thread length in characters (len (ch)), number of messages in thread branch, the number of actual topic changes in the thread (# Changes), the number of topic changes found (# Found), and the number found in the correct location (Cor. Loc). P, R, and F give the precision, recall and F-score, as defined in 3.1.7 . . 42

4.10 DBSCAN results for (N=10, T=5, E=6, MP=4). Results show the Thread ID, thread length in characters (len (ch)), number of messages in thread branch, the number of actual topic changes in the thread (# Changes), the number of topic changes found (# Found), and the number found in the correct location (Cor. Loc). P, R, and F give the precision, recall and F-score, as defined in 3.1.7 . . 43

4.11 DBSCAN results for (N=10, T=5, E=10, MP=6). Results show the Thread ID, thread length in characters (len (ch)), number of messages in thread branch, the number of actual topic changes in the thread (# Changes), the number of topic changes found (# Found), and the number found in the correct location (Cor. Loc). P, R, and F give the precision, recall and F-score, as defined in 3.1.7 . . 43

4.12 DBSCAN results for (N=25, T=51, E=15, MP=10). Results show the Thread ID, thread length in characters (len (ch)), number of messages in thread branch, the number of actual topic changes in the thread (# Changes), the number of topic changes found (# Found), and the number found in the correct location (Cor. Loc). P, R, and F give the precision, recall and F-score, as defined in 3.1.7 44

4.13 DBSCAN results for (N=25, T=51, E=5, MP=3). Results show the Thread ID, thread length in characters (len (ch)), number of messages in thread branch, the number of actual topic changes in the thread (# Changes), the number of topic changes found (# Found), and the number found in the correct location (Cor. Loc). P, R, and F give the precision, recall and F-score, as defined in 3.1.7 . . 44

4.14 Sliding window results for window size = 20. Results show the Thread ID, thread length in characters (len (ch)), number of messages in thread branch, the number of actual topic changes in the thread (# Changes), the number of topic changes found (# Found), and the number found in the correct location (Cor. Loc). P, R, and F give the precision, recall and F-score, as defined in 3.1.7 . . 45

4.15 Sliding window results for window size = 40. Results show the Thread ID, thread length in characters (len (ch)), number of messages in thread branch, the number of actual topic changes in the thread (# Changes), the number of topic changes found (# Found), and the number found in the correct location (Cor. Loc). P, R, and F give the precision, recall and F-score, as defined in 3.1.7 . . 45

4.16 Sliding window results for window size = 100. Results show the Thread ID, thread length in characters (len (ch)), number of messages in thread branch, the number of actual topic changes in the thread (# Changes), the number of topic changes found (# Found), and the number found in the correct location (Cor. Loc). P, R, and F give the precision, recall and F-score, as defined in 3.1.7 . . 46

Acknowledgements

To Robbie, thank you for being my partner. You have supported me and this work as a spouse, colleague, and best friend.

To Will, thank you for inspiring the passion of empiricism in me.

To Matt, thank you for being a generous mentor and wonderful friend.

To Marco Draeger, Jenny Tam, Jon Durham, and the rest of the NLP lab, thanks for all your help.

Thank you to Craig Martell, for pushing me when I needed it most. You were right about the things that mattered.

Thank you to my parents for leading by example, and always supporting me in creating my own path.

Finally, to General Pete Worden, for taking a bunch of young troublemakers under his wing and helping us make a careers out of it.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

1.1 Introduction

This thesis applies Latent Dirichlet Allocation (LDA) to the problem of topic and topic change in conversational threads using e-mail. Using LDA, probabilistic models are built for the topics within the corpus, and these topics are used both to cluster e-mails with their original threads, as well as to study where these topics change within threads.

We demonstrate that LDA can be used to successfully classify raw e-mail messages with threads to which they belong, and compare the results with those for processed threads, where quoted and reply text have been removed.

We then present two new, unsupervised techniques for identifying topic change in e-mail. The first takes keywords identified by the LDA algorithm and clusters them to identify topics in threaded conversations. Topic changes are then by definition the transitions between clusters. The second technique uses a sliding window over each thread. For each window, the current topic is calculated using the LDA word-topic weights, and a note is made when this topic changes.

1.2 Motivation

The resolution, accuracy and availability of sociological information is increasing at a rapid rate. So, too, is our ability to quantify that data. Never in history have we had such fine grained, quantitative measures of individuals' actions across such vast and diverse swaths of people. The consistency and benevolent nature of this data goes orders of magnitude beyond what was possible even ten years ago. To have compatible and consistent data sets across large groups of people has historically been messy and difficult, if not impossible.

Today, not only has this type of data become more available, it has become so ubiquitous that it requires us to develop new approaches to studying them. Machine learning algorithms, including probabilistic topic models, are a promising approach.

Mining through these large data collections to detect consistent patterns and trends can give us empirically verifiable data about the nature of human social dynamics and interactions. Mining

through them for anomalies and deviations from some reference point can suggest events or individuals worthy of further study.

For example, it is a well known problem in digital forensics that investigators are often given large hard drives to analyze, with little or no indication of where to start or what the important documents are. A personal hard drive in 2009 can easily be half a terabyte in size. Much of this data can be natural language content, with conversational content interspersed throughout. Without statistical methods to handle all this data, scientists and investigators alike are lost.

E-Mail specifically, is a rich source of information about the dynamics of information flow in social networks. It is inherently structured, with a variety of time, author, and content-related metadata fields. Its increase in availability on phones and online has increased its prevalence as an easy and accessible communications medium for the full range of online tasks, from communications to grocery lists and event planning. Storage trends in reliability and affordability mean people are retaining more and more of their e-mail, and for longer periods of time.

E-Mail presents an analytical challenge different from longer documents such as articles, reports or books. E-Mail bodies can vary in length from a few words to multiple paragraphs. Their often terse nature can push the limits of most content- and topic-analysis algorithms in use today.

Interestingly, e-mail might not be such a rich source of information for long. As technology-based communications diversify into more customized and better suited platforms such as instant messenger, social networking sites, and wikis, we may no longer have the luxury of a single, de-facto platform for the exchange of content between individuals. We should take the opportunity to study e-mail corpora now, while they are still in widespread use.

Motivated by a desire to better understand characteristic patterns of human interaction, this thesis focuses on the question of how conversations evolve. Using e-mail for its threaded conversational nature, topic and topic change are studied are studied by examining patterns of word usage in e-mails. Specifically, this work uses new and emerging techniques in data mining and machine learning to show that we can build accurate models of topics in e-mail threads, using state-of-the-art probabilistic techniques. It then extends these techniques to the problem of identifying topic change within threads, providing a baseline for what is possible with current methodologies, and identifying directions for future work in this area.

1.3 Applications

In Marti Hearst's paper "Text Tiling" (discussed further in Chapter 2), she points out that topic change can be seen as an inverted approach to topic detection. If we know where topic changes, we also know the boundaries of topics [13]. Beyond a study of topic boundaries, the nature and frequency of topic change is interesting in its own right, giving us insight into the dynamics of human communications.

Topic change is useful for understanding the ways that content or opinion change over time in personal relationships, public sentiment, news coverage, and the evolution of interests within populations. Digital forensic investigations are often interested in the point at which the nature or content of a relationship changes. The ability to detect topic change would support investigations of sexual predators, where a conversation often starts out platonic and then turns sexual [21], as well as investigations into the techniques of recruitment for criminal or religious activity.

If social networking sites can identify the point at which new topics emerge in individual or aggregate discussions, they can make better recommendations, and even serve better advertisements.

Companies would often like greater insight into what their employees are talking about. Are discussions beginning on the topic of work, and then frequently evolving towards more social topics? Are managers confusing or distracting their group by assigning too many projects at once, or changing team goals too often? Could this be correlated to successfully or unsuccessfully managed projects?

Looking at the more subtle aspects of topic change, the ability to observe changes from positive to negative sentiments about a topic, or discussion of one policy issue moving into another (for example, as Congress' schedule changes) would be very interesting. This has huge potential to help policy makers (in fact, all of us) understand public sentiment. The ability to correlate the actions of leadership in any group, with the presence, speed and nature of its constituency's reactions, or the path of a topic's flow through different demographics, could greatly aid in improving feedback loops, and forming more effective policies.

Topic change can also be used to study how our understanding or priorities *within* certain topics change. Blei applied dynamic topic models applied to 100 years of the journal *Science*, to see how our understanding of topics such as quantum physics and neuroscience had changed

[2]. Similarly, many government organizations such as the National Academy of Sciences, and the National Aeronautics and Space Administration (NASA), undertake decadal surveys of research priorities. The study of topic change can help us to see not only how those priorities have changed between surveys, but could also be used to find differences between identified and implemented priorities, through documents associated with actual missions or studies undertaken. Automated techniques are useful here, both because of the quantity of data, and because automated techniques are objective in a way that can be difficult for humans.

More theoretical research could look for characteristic patterns in the evolution of ideas, opinions or populations, captured in text over time. Sociologically, the ability to measure patterns of communications across cultures, age groups, dispositions, or mental conditions could all provide insights into ways to improve communication or simply improve insights in their nature. How often do topics or popular opinion change within specific realms, generations, or nations? To what degree does the average conversation change topic? Do conversations frequently change topic abruptly, or in a slow and meandering fashion? When they do or do not, what does it imply about the participants, their relationships, or the underlying topics themselves?

1.4 Relationship to Space Exploration

As this is NASA-sponsored research, we touch briefly upon how this work can support ongoing efforts to advance exploration and settlement of the solar system.

Conversational document clustering can be applied to transcripts of verbal communications, and written communications, of astronauts on board the space station. Psychological and sociological studies focused on stress and quality of life factors, collaboration dynamics, and team effectiveness could benefit from the ability to tie together threaded conversations. Similarly, insight into the dynamics of topic change, as described above, could also support our insight into requirements for successful team dynamics, in space or otherwise. On a more immediate level, studies of organization priorities and undertakings, as mentioned above, could help institutions begin to understand where discrepancies arise between intention and implementation. Like many large organizations, NASA is no stranger to these questions. The larger and more distributed a group (and NASA has over 50,000 people across 10 centers [8]), the harder it can be to make implementation match intention. More people tends to mean more data, and as our understanding of how to measure topic change develops, we can actually examine where and how these shifts take place.

1.5 Outline of this Thesis

In Chapter 2, an overview of supporting and related work is provided. Chapter 3 goes into the mathematics and theory behind the techniques and concepts used in the experiments for this thesis, and describes the data processing done to the corpus. Chapter 4 outlines the experiments and their results, and Chapter 5 discusses the implications of these results and ideas for future work. Chapter 6 offers conclusions.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Prior and Related Work

Topic detection and the automated detection of topic change touches on a number of different areas in natural language processing, machine learning, and visualization techniques.

Topic detection has been around in varying forms since the earliest work in natural language processing. Techniques such as word-sense disambiguation, clustering, cosine similarity, and basic word-counting have been applied with reasonable success to the topical problems of automatic summarization, sub-topic clustering, probabilistic topic modeling, and keyword identification.

Topic change as a phenomenon in its own right has been studied in relatively fewer areas, although some researchers, such as Hearst [13], propose the detection of topic change as a reformulation of the problem of topic detection. Others, such as Blei and Lafferty [2], have studied the evolution of topics over decadal time spans as a way to observe changing norms, practices, and beliefs as reflected in popular scientific literature.

A variety of more visual techniques have also been applied to natural language corpora, in attempts to capture and represent intuitive notions of conversational evolution, which is another way to approach detection of topic change. These are described below.

2.1 Topic Detection

At its core, topic change is heavily related to topic detection. State of the art in topic detection is currently rooted in probabilistic models which hypothesize a *latent* topic space, manifested in the documents of a corpus; that is, that there is a specific set of topics represented by the documents, and each word w has a probability of belonging to each of the topics, z, with a given probability. That is,

$$P(w_i) = \sum_{j=1}^{T} P(w_i|z_i = j)P(z_i = j) \tag{2.1}$$

(This equation is developed more fully in Section **??**). Initial work in probabilistic topic modeling was called Latent Semantic Analysis (LSA), which applied Singular Value Decomposition

(SVD) to term frequency counts in order to identify documents likely to be of the same topic [6]. Probabilistic Latent Semantic Indexing (PLSI) [14] built on LSA by associating a latent topic variable with each word, and postulating that the probability of a given word in a document is actually given by summing over the probabilities of that word across each latent topic, times the probability of each topic given a document.

By introducing the latent topic variable as a probability distribution, P'S replaced SVD with probabilistic mixture models. The output of PLSI is a set of mixing weights, representing the contributions of the various topics to the document.

By representing topics as probability distributions, PLSI provides an intuitive notion of topics not as discrete objects which begin and end on a boundary, but as distributions which can, and do, mix in with other topics.

However, while PLSI models words using probabilities across topics, there is no generative model of how these probabilities arise, that is, of the topics themselves. The Expectation-Maximization algorithm used in PLSI to estimate topic likelihood is, partially as a result, prone to overfitting, and the number of parameters can grow linearly with the size of the corpus [3, p.2].

Blei, Ng, and Jordan attempted to address these shortcomings with a technique called Latent Dirichlet Allocation (LDA). This technique represents documents as random probability mixtures over latent topics, but each topic is itself represented as a distribution over words [3]. A prior is assumed on the topics, drawn from a Dirichlet distribution. The parameters of this Dirichlet are known as the mixing weights.

BuzzTrack [4] uses cosine similarity between messages as one of the features of its topic detection and tracking system. Damashek [5] uses cosine similarity to cluster documents by removing the average of a document set from each document to be clustered, and then clustering on the remaining values. He finds that good, language-independent results are obtained. However, his technique is sensitive to *any* differences in the document bodies, and these can be sentimental as well as topic differences.

2.2 Topic Change

Two papers focusing specifically on the notion of topic *change* are worth noting here. The first is Marti Hearst's "TextTiling" paper [13], which analyzes documents on a paragraph-by-paragraph

basis and uses "patterns of lexical co-occurrence and distribution" to detect transition from one topic to the next. Hearst removes stop words and applies stemming, and then evaluates two metrics to formalize the notion of co-occurrence. The first metric uses a normalized dot product (or cosine similarity) between two adjacent blocks, where blocks are approximately paragraph-length. The second metric is called "vocabulary introduction", and is calculated as the ratio of new words in a given interval, to the length of that interval. Hearst's results are comparable to or better than state of the art at the time. Hearst's work is applied to journal articles with a single set of authors. This differs substantially from the conversational corpus we use here.

Blei and Lafferty [2] extend Latent Dirichlet Allocation (LDA) with a temporal element in order to study the evolution of topics over time. Specifically, LDA makes no assumption about the sequential aspect of documents in a corpus; however, in this work, each year is modelled as a set of topics, which are themselves a function of the set of topics in the previous year. They apply this technique to 120 years of archives of *Science*. At ten year intervals, they compare keywords from specific (pre-defined) categories such as Neuroscience and Quantum Physics, and the result is a study of how these keywords change on decadal time frames. Blei and Lafferty apply their dynamic topic models to predict the time-evolution of topics, and their results show that improved predictive accuracy can be obtained with these dynamic rather than static topic models.

Blei and Lafferty's work uses set intervals on which they study changes in topic contents, whereas we instead attempt to automatically detect when topic change occurs. Their corpus contains a pre-defined list of categories, or high level topics, within which they focus their analysis of changing keywords. Finally, they examine ten year increments of a larger corpus, while we examine changes on the hourly or daily time frame, as represented by mere sentences or paragraphs.

In this work, we explore e-mail, which is terse and informal. It is addressed to an internal audience (those copied on the e-mail). In addition, because of the conversational aspect, there are often new elements of a thread which lexically have little or nothing in common with the previous sentence or e-mail, but which refer to the same topic. These factors provide a rather different context for detecting topic change.

2.3 Conversation Flow

Innovative methods for representing conversation flow are the focus of many studies on e-mail, as well as blog comments and revision controlled documents. Many of these studies develop techniques to visually highlight important elements of conversations, and use the viewer as the classifier. Their success or failure is judged on the assessment of the viewer. In this sense, most of them do not represent algorithms which can be automated, as we are exploring here, but they do represent a legitimate approach to representing the evolution of topics and conversation flow.

Two particularly interesting efforts in this regard can be found in the techniques of *history flow* and *theme river*.

History flow [22] visualizes changes in the life cycle of a document by representing each revision as a vertical line, and contributions by each author as individually coloured lines between these revisions. The thickness and location of each author's line corresponds to the number of words, and location in the document, respectively. If a new contribution is made, a new line is started; if one is removed, that line is terminated.

History flow has been used to visualize edits over time of Wikipedia articles. It demonstrates interesting, consistent visual patterns over time corresponding to specific behaviours such as edit wars and vandalism.

Theme river [12] is a visualization technique which hand selects a specific set of key words in a series of documents, and represents their relative frequency of occurrence as the thickness of a smoothed line. This line "flows" along a time line of the documents in question. The thickness is meant to intuitively convey those words in the set which appears frequently in any given document.

Theme river is visually appealing and intuitive in its representation. Although the version implemented in the paper requires much supervision and hand labeling, one could imagine modifications to such a system which would automatically determine keywords, and track their role as a conversation evolved. These techniques are an alternative way of exploring the notion of conversational evolution.

2.4 E-Mail

E-Mail corpora have been used extensively for studies as diverse as thread prediction, authorship studies, role identification, spam filtering, topic detection, information synthesis, keyword

extraction, and many more. We will not review all applications of e-mail research here; however, we touch briefly on the Enron corpus [15] to discuss why it was not used.

By far, the Enron corpus is the most widely used corpus for research in the area of e-mail [15]. It consists of approximately half a million e-mails from 150 users, with attachments removed. Although this corpus presents a valuable option for many research areas, there are two principle reasons why it was not used here (also identified by [4]):

- Ground Truth – working with other peoples' e-mails provides a difficult reference point for ground truth, in terms of topic identification, relevance or meaning. In this set of experiments, using the author's personal e-mail enabled more knowledgeable and accurate interpretation of topics and topic change.
- Metadata – The experiments in this thesis all revolve around the e-mail thread, which are programmatically re-constructed from individual e-mails using the The In-Reply-To header field populated by most e-mail systems. The vast majority of e-mails in the Enron corpus do not have this header field, which would have prevented thread-based analysis.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Technical Concepts and Data Processing

In this chapter we document the techniques, concepts, and technical approaches used in the experiments undertaken for this thesis. As the intended audience spans the digital forensics and natural language processing communities, as well as graduate students in other fields, certain foundational terms and concepts are covered.

3.1 Technical Concepts

3.1.1 Natural Language Processing

In Natural Language Processing (NLP), researchers attempt to give structure to unstructured, or natural, language documents, in order to build tools or algorithms which analyze patterns and meaning in the content. This is typically done by splitting documents into words, and analyzing those words either individually (the 'bag-of-words' approach, see below), or as n-grams, sentences, or paragraphs. The process of identifying word boundaries is called *tokenization*, and the resulting objects are formally called *tokens*.

The term 'bag-of-words' is used to describe an approach to text processing where the words are treated as isolated entities, without regard to their immediate context or order. Conceptually, it's as though the words in a document were thrown into a bag; but more importantly, a bag is a technical term that, as opposed to a set, allows for duplication of tokens. For many applications this is a useful simplifying approach. This thesis uses the bag-of-words approach for topic modeling in several of the experiments. N-grams are ordered sequences of n words or characters, and they give additional context to words (or characters) in a document. Because the possible number of n-grams in a document with vocabulary of size V is V^n, the number of possible n-grams for a given vocabulary is (relatively) huge, while the number of actual n-grams in a document is low compared to this possibility space. Natural Language data sets, for this reason, are often referred to as 'sparse.' Naturally, as n increases, so does the sparsity of the coverage. Because of this sparsity, the probability of any specific n-gram is very low, and thus its presence can be a strong indicator of a feature (such as a topic or author). However, greater sparsity may come hand in hand with over-training, and selection of the right n is typically a trade off between accuracy and coverage–as is the case with most statistical techniques.

Whatever the basic unit of analysis, many NLP techniques involve creation of some kind of

vector space model. Each word (or n-gram) in the vocabulary represents a dimension, and each document can then be represented by a vector. The dimensions correspond to the words or n-grams therein, and the value of each dimension corresponds to the number of occurrences of that object. In many cases, the value for a given dimension will be 0 if it has not occurred in that particular document.

For example, consider the sentence, "Government data transparency is important for government to function well." tokenized on white space boundaries.

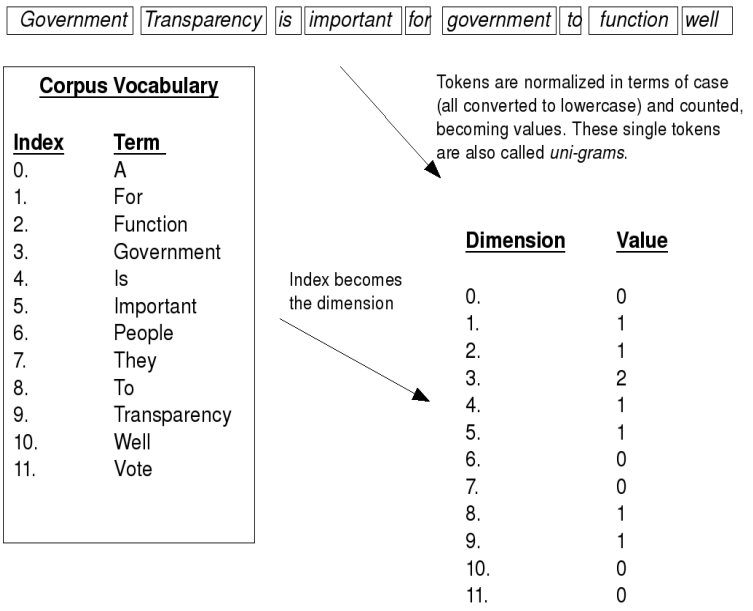

Figure 3.1: Basic word tokenization and vector-space representation for unigrams.

By converting documents to a vector space representations, the tools of geometry and algebra can be applied, and questions of difference and distance between documents become meaningful.

3.1.2 Supervised and Unsupervised Learning

There are two basic categories of machine learning algorithm, *supervised* and *unsupervised*.

Supervised learning involves a problem where the number of classes or groups into which the input data is being sorted, is known ahead of time, or determined at some point in the analysis. Supervised learning asks, "to which of these n groups does this record or data point belong?"

Supervised learning is often associated with classification problems. It is considered *supervised* in the sense that the learning task has examples of correct patterns to work with, in order to build its model. Supervised learning is used in the e-mail classification experiment in this thesis.

In contrast, unsupervised learning involves determining as part of the algorithm, the number of classes or groups of the final output, as well as which records fit into which classes. It is *unsupervised* in the sense that the algorithm has no set of known correct examples guiding the development of a model. Unsupervised learning is often associated with clustering problems. Unsupervised learning is used for the topic change experiment in this thesis.

3.1.3 Cross Validation, Test and Training Data

In machine learning, data is needed to train a model, but once that model has been built, data is also needed to test the quality of that model. N-fold cross-validation is the practice of dividing a dataset into N equal-sized subsets, and then iteratively reserving one, training on the remaining $N - 1$ subsets, and testing on the reserved subset. A typical number for N is 10%.

Our corpus contained a large number of threads, with a large number of messages. A 20/80 split was used between our test data and training data.

3.1.4 Latent Dirichlet Allocation

As introduced in Chapter 2, Latent Dirichlet Allocation (LDA) is a technique which models a natural language corpus as a probabilistic distribution over topics.

Before defining LDA itself, recall these mathematical concepts from probability theory:

A conjugate prior is a prior which results in a posterior probability distribution of the same algebraic form, or family, as the prior.

A multinomial distribution of order k is one in which each time a measurement is made, exactly one of k possible outcomes occurs. Multinomial distributions are also sometimes referred to as categorical distributions, where there are k categories. The most popular form of multinomial distribution is, of course, the binomial distribution.

Each topic has a probability distribution over the documents in the corpus, and each word has a probability distribution over the topics in the documents. These distributions are multinomial

distributions, because each time a selection is made, exactly one topic or word is chosen from the possibility set.

The main question in probabilistic topic models, is how to model these distributions, and what the assumptions, or priors, will be about those distributions. The following derivation follows closely what is presented in [19].

The probability distribution over words in a given document for T topics is

$$P(w_i) = \sum_{j=1}^{T} P(w_i|z_i = j)P(z_i = j) \tag{3.1}$$

For a given topic j, let $\phi^{(j)} = P(w|z = j)$ be the distribution over words in the corpus for topic j; similarly, $\theta^{(d)} = P(z)$ is the distribution over topics for a specific document, $d \in D$, where D is the total set of documents in the corpus. Then (3.1) can be written as

$$P(w_i) = \sum_{j=1}^{T} \phi^{(j)}\theta^{(d)} \tag{3.2}$$

ϕ and θ are referred to as the mixture weights of the words and topics, respectively. These parameters indicate which words are important for which topics, and which topics are important for which documents.

To give a starting point for determining these mixing weights, Blei et al. [3] apply a prior to the topics in the form of a Dirichlet distribution. The Dirichlet distribution is a conjugate prior for the topic mixing weights θ. For a model with T topics, the T-dimensional Dirichlet is a distribution over the set of possible probability distributions $p = (p_1, \ldots, p_T)$ for the topics, and is given by

$$Dir(\alpha_1, \ldots, \alpha_T) = \frac{\Gamma(\sum_j \alpha_j)}{\Pi_j \Gamma(\alpha_j)} \Pi_{j=1}^{T} p_j^{\alpha_j - 1} \tag{3.3}$$

The parameters $\alpha_1, \ldots, \alpha_T$ are considered *hyperparameters* for the topic model itself. In practice, symmetric hyperparameters are used, such that $\alpha_1 = \alpha_2 = \ldots = \alpha_T = \alpha$.

Figure 3.2 shows a 2-dimensional representation of the probability space for 3 topics following a Dirichlet distribution. The triangle shape in the figure is a geometric notion called a simplex. The simplex is a coordinate system for probability distributions; each point p in the simplex is a T-tuple of the probabilities p_j for each topic $j \in T$, and $\sum_j p_j = 1$– that is, as required, the sum over the probabilities for each of the topics is 1. One can think about the simplex as the $(n-1)$-dimensional region that connects the basis vectors for n-space. So for 3-space, we would have the 2-dimensional triangle between $(1,0,0)$, $(0,1,0)$, and $(0,0,1)$.

The hyperparameter α affects the smoothing of the topics across the simples. Higher alpha has a squeezing effect, focusing the distributions around the center of the simplex, and therefore resulting in greater smoothing or similarity between the different p_js. In practice, α is generally set to ¡ 1, which pushes the modes of the Dirichlet to the corners of the simplex, leading to greater distinction between the different topics.

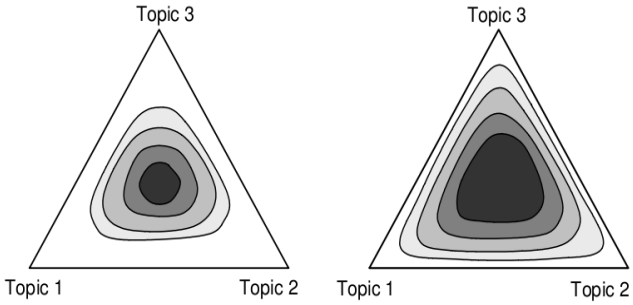

Figure 3.2: Symmetric Dirichlet distribution for three topics on a 2-dimensional simplex. Darker colours indicate higher probability. Left: $\alpha = 4$; right: $\alpha = 2$. Figure and caption from [19, p.5].

Steyvers and Griffiths [9] [10] [11] extend this model by applying a Dirichlet prior $Dir(\beta)$ to the mixing weights of the words over topics, ϕ, as well. Thus, there are now two hyperparameters to the model: α and β.

For the implementation, instead of estimating the latent hyperparameters ϕ and θ, the algorithm directly estimates the probabilities for each topic $j \in z$ directly, using a process called Gibbs sampling. Once the posterior for z has been estimated, ϕ and θ can also be estimated.

In Gibbs sampling, each word or token in the corpus is given a probability of arising from a given topic, as a function of the topic probabilities for all other words in the corpus. This

conditional probability distribution is derived in [20] and given in [19] as follows:

$$P(z_i = j|\mathbf{z}_{-1}, w_i, d_i, \cdot) \propto \frac{C^{WT}_{w_ij} + \beta}{\sum_{w=1}^{W} C^{WT}_{w_ij} + W\beta} \frac{C^{DT}_{d_ij} + \alpha}{\sum_{t=1}^{T} C^{DT}_{d_ij} + T\alpha} \tag{3.4}$$

C^{WT} and C^{DT} are WxT- and DxT-dimensional matrices, respectively, of token counts. The first matrix contains the number of time a word w_i is assigned to topic j; the second matrix contains the number of times topic j is assigned to a word token in document d. In practice, equation (3.4) is normalized by the number of topics, T.

The Gibbs sampling process itself has two phases, an initial so-called *burn-in* period, during which samples must be discarded, and post-burn-in, during which samples begin to converge on the true posterior distribution more accurately. The process of Gibbs sampling begins by assigning a random topic to each word token. Then, a new topic (where, recall, a topic in this case is simply another probability distribution based on the Dirichlet prior) is sampled from equation (3.4), and the count matrices are updated based on the new topic assignment. *Every sample* performs a topic assignment for all N word tokens in the corpus.

Because the initial topic assignments are random, and because new topic assignments are samples from a probability distribution, multiple Gibbs samples from after the burn-in period must be obtained and combined, to generate a representative sample.

The output of the LDA algorithm using Gibbs sampling is a set of weights for each word in the corpus, for each topic. That is, for a corpus of N words, each topic will contain N words, with a corresponding weight indicating its likelihood of being drawn from that topic. A smoothing process means that a word never has an absolute zero probability of being drawn from any topic. These outputs can be used as inputs to other algorithms, or can be used directly to estimate the topic probabilities of new documents.

3.1.5 Distance and Similarity Measures

Four measures of similarity or distance are used in our experiments, falling into two categories– distance metrics, and entropic measures.

To be considered a true metric, a distance measure must satisfy the properties that, for three

vectors x, y, and z,

$$d(x, y) = 0 \iff x = y$$
$$d(x, z) < d(x, y) + d(y, z)$$

The latter condition is known as the triangle inequality.

The euclidean distance is probably the most familiar notion of distance, and is the only true distance metric used. Recall that for two n-dimensional vectors x and y it is expressed as

$$d = \sqrt{(y_1 - x_1)^2 + \ldots + (y_n - x_n)^2} \tag{3.5}$$

Euclidean distance has a range of $[-\infty, +\infty]$.

Cosine similarity is a measure of similarity between two vectors. It is calculated by taking the cosine of the angle between two feature vectors. This is a rather intuitive measure of similarity: when two vectors are exactly the same, the angle between them is 0, and the cosine of the angle between them is 1; when the vectors are orthogonal, the cosine value is 0. The cosine similarity is given by the quotient of the dot product between the two vectors, with the product of their length. If x and y are the input vectors, the cosine similarity is given as

$$\cos\theta = \frac{x \cdot y}{\|x\|\|y\|} \tag{3.6}$$

As can be seen from (3.6), the measure is inherently normalized, and its range is $[-1, 1]$. Because of this normalization, two vectors in the same direction with different lengths, will have a similarity value of almost 1, but will have a euclidean distance equal to the difference in their lengths, which in general can be arbitrarily large. One can see, then, why cosine similarity is a useful tool for comparing the similarity of documents, since we would indeed likely consider two documents of different lengths with the same words 'similar'.

For the e-mail classification experiment, classification decisions are based on comparison of the distribution of topic probabilities between a set of potential threads and a test e-mail. As a result, we also calculate the Kullback-Leibler (KL)- and Shannon-divergence. Both are entropic

measures of difference between probability distributions.

KL-divergence between P and Q is an asymmetric measure of the amount of information, or number of bits, needed to encode distribution P based on distribution Q. It is calculated in the following way:

$$KL(P,Q) = \sum_i P(i) \log P(i)Q(i) \tag{3.7}$$

The Shannon divergence is a symmetrized version of the KL-divergence. It is essentially the average of the KL-Divergence in both directions:

$$Shannon(P,Q) = Shannon(Q,P) = \tfrac{1}{2}KL(P, \tfrac{1}{2}(P+Q)) + \tfrac{1}{2}KL(Q, \tfrac{1}{2}(P+Q)) \tag{3.8}$$

Experimental results are calculated using all four difference metrics, and compared in detail in Chapter 5.

3.1.6 DBSCAN

DBSCAN stands for Density-Based Spatial Clustering of Applications with Noise. As the name suggests, it is a density-based clustering algorithm which efficiently discovers clusters of arbitrary shape. DBSCAN also allows for certain points to be deemed noise, and therefor not allocated to any cluster [7].

DBSCAN is used in the automated detection of topic change experiment, because we do not want to make assumptions a priori about the number of topics there are in a thread.

To discover clusters, the algorithm looks for points which have a minimum number of neighbouring points, min_pts, within some radius ϵ (i.e., sets of points with a certain density). Any distance function can be used to compute the radius, although euclidean space is used here.

A key observation of the DBSCAN algorithm is that there are two kinds of points in a cluster: core points and boundary points. Core points will contain the requisite neighbour density, but boundary points will not.

Thus DBSCAN demands that all points within a cluster are *density-reachable* from one another.

Instead of requiring every point in the cluster to have min_pts within its ϵ-radius, the requirement of density reachability instead requires that every point in the cluster have a *neighbour* within ϵ that has min_pts within its radius. That is, for every point q in a cluster C, there must be another point p within the epsilon neighbourhood of q that has min_pts in its neighbourhood. p is then said to be directly density-reachable from q, and points reachable in a chain of *directly*-density-connected points are considered just plain density reachable.

Finally, there may be some points in a cluster which are not density-reachable, but which are connected via a common density-reachable point. These are called *density-connected*.

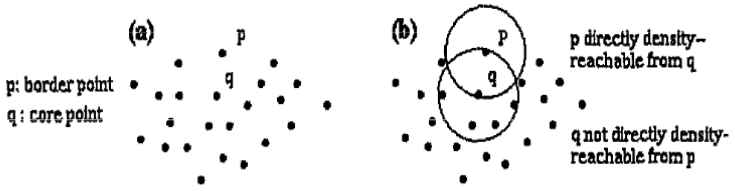

Figure 3.3: Illustration of points that would be considered border points and core points in the DBSCAN algorithm. Figure from [7, p.3].

All density-reachable and density-connected points make up a cluster. More formally, let D be a set of data points. A cluster, defined with respect to ϵ and min_pts is a non-empty subset C of D satisfying the following conditions:

1. $\forall p, q$ if $p \in C$ and q is density-reachable from p with respect to ϵ and min_pts, then $q \in C$.
2. $\forall p, q \in C, p$ is (at least) density-connected to q via ϵ and min_pts.
3. Any point $p \in D$ and $\notin C_1 \ldots C_n$, where $C_1 \ldots C_n$ are the clusters of the data set, is considered noise.

Figure 3.4 shows an example of both density-reachable and density-connected points. For topic change, we use DBSCAN to identify clusters of keywords forming a topic, and most importantly, where the boundaries of those clusters are. Boundaries are affected by the input parameters to the algorithm, and are the key element in identifying transition points, or topic changes.

Appendix A contains the code for this algorithm, and current links to its availability online.

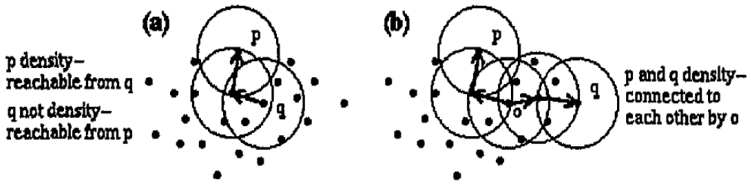

Figure 3.4: DBSCAN density-reachable and density-connected points. Figure from [7, p.3]

3.1.7 Topic Change

In order to give some quantitative measure of the performance of the topic change algorithms, the traditional measures of precision and recall were used– being calculated as functions of true positives (tp), false positives (fp), and false negatives (fn)– with very slight modification. For a given thread, we care both about the number of changes detected, and the accuracy of the locations of those changes. For a thread with N actual topic changes, if D changes were detected, and L of those changes were in the correct location, then the precision is defined as:

$$p = \begin{cases} \frac{tp}{tp+fp} = \frac{L}{L+(D-L)} = \frac{L}{D} & \text{if } D \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

False positives are the number of topic changes detected that were not actually topic changes. Recall is defined as

$$r = \begin{cases} \frac{tp}{tp+fn} = \frac{L}{L+(N-D)} & \text{if } N > D \\ 0 & \text{if } L = 0 \\ 1 & \text{otherwise} \end{cases} \tag{3.10}$$

The notion of false negative corresponds to the number of topics that were missed–those that were considered negatives but were actually positives. Consider a thread that contains one topic change. If 50 topic changes were detected, including one in the correct location, then the precision is low, because there were many false positives. However, the recall would be 1, because there were no false negatives–that is, all the changes that exist were detected (and in the correct location).

Similarly, if a thread has 2 topic changes, and 2 changes were identified, but neither in the correct location, then both precision and recall would be 0.

It is worth noting that true negative is not correspondingly well defined here. One could think of the number of true negatives as the number of words, sentences, or paragraphs during which a topic change does not take place, but it is somewhat different from the traditional notion of true negative.

The F-score is calculated in the usual way:

$$F\text{-}score = \begin{cases} \frac{2pr}{p+r} & \text{if } p > 0 \text{ and } r > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.11}$$

The F-score is used as a measure of accuracy because it is a special form of the harmonic mean. The consequence of this algebraic form is that neither P nor R can be inflated at the other's expense, without compromising the total F-score [16].

3.2 Data Processing

The corpus used for this thesis was 4.9GB of the author's personal e-mail over the course of approximately 3 years.

Although these experiments use e-mail for its rich social value, there is a large amount of other content which must be accounted for when processing arbitrary e-mail data sets. Discussion lists, announcements, calendar invites, administrivia, company-wide broadcasts, and notifications from reminder services. Further, some services such as Gmail store chat logs as e-mails; some people might use e-mail as a way to send themselves reminders, or even as a file store. This is all noise from the perspective of our research.

To remove the vast majority of these noisy e-mails, only threads of length 5 or longer were retained. 5 was chosen because we are interested in threads where the topic has some time to develop, and where there is a meaningful conversational (back-and-forth) component.

After removing shorter threads, empty messages, and messages with only attachments or with unrecognized encodings, there were 2168 threads remaining.

3.2.1 Experimental Overview

Figure 3.5 shows a high-level overview of the steps taken in order to pre-process the e-mail data, and how and in what order the various techniques described herein are applied.

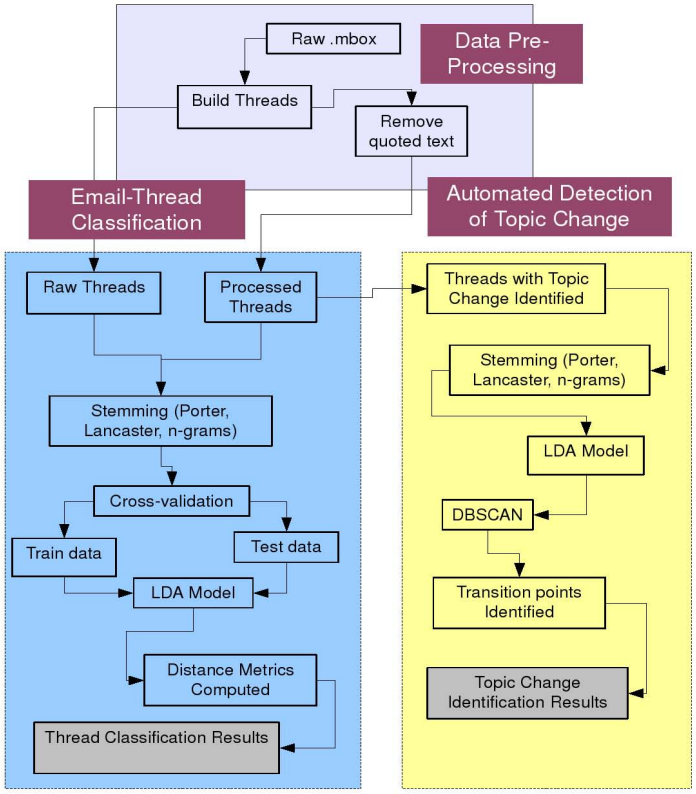

Figure 3.5: High-level overview of steps taken for data pre-processing, and both experiments.

3.2.2 E-Mail Processing

Corpus messages were in Mbox format. Although Mbox format can come in several different flavours, generally speaking it is a plain text electronic mailbox format which stores all messages within a folder in a single file. Messages are separated by a blank line and their beginning is delimited by the word `From_` (note the space after the word, underlined for emphasis).

Mbox files were parsed using Python's e-mail-handling modules, and converted to internal Message objects which stored the headers, message bodies, and in-reply-to header field, if it was present (this field is described more in Section 3.2.3, below. Message bodies were extracted using the `Content-Type` header field, which describes the MIME-type formats contained in the message. Often times, a message will be `multipart`, containing a plain text formatted version of the message along with HTML or other encodings.

However, some messages are not multipart, nor do they have a plain text component. Some of these are HTML only. The Content-type 'Message' is one of these. In this case, the message can be encapsulated, along with possible error messages from servers, digests, forwarded messages, etc.

For our purposes, anything not plain text- or HTML-formatted was discarded. The plain text body was selected if it existed, otherwise, if an HTML version was present, it was used, and the HTML removed separately. Message bodies were stored in UTF-8, replacing any unrecognized characters with the Unicode replacement character \uFFFD.

3.2.3 Thread Extraction

A thread is a series of e-mails which are related to one another through the messages they reply to. A thread is a tree-type data structure because multiple e-mails may be in response to the same e-mail, and messages may not necessarily be in response to the latest message in a thread.

Threads were reconstructed using the `message-id` header field, and `in-reply-to` header fields of e-mail messages.

The `Message-id` field contains a globally unique identifier typically made up of a message hash followed by an '@' symbol and the mail server domain. For example: `3cb0e8e0610091234q5affb09fq2969c9ca2a051c17@mail.gmail.com`.

The `in-reply-to` field also contains a message ID; that ID is of the message which the current message is, not surprisingly, in reply to. This field should not to confused with the `reply-to` header field, which is a user-specified preferred e-mail address for message reply.

Two passes are made over the messages to reconstruct the threads. During the first pass, if a message's `in-reply-to` header field matches the `message-id` field of another message, that message is added to the thread. If a message either does not have an `in-reply-to` header field, or that field does not match any messages already in a thread, a new thread is created.

The second pass is a consolidation pass, since as threads are reconstructed, it may turn out that the initial message of a thread was in fact in reply to a message that came later in the processing queue. If this is the case, the two threads are consolidated.

For each thread, we then identify each possible branch through that thread. Because we are interested in studying topic change over the course of a thread, how to handle the branching

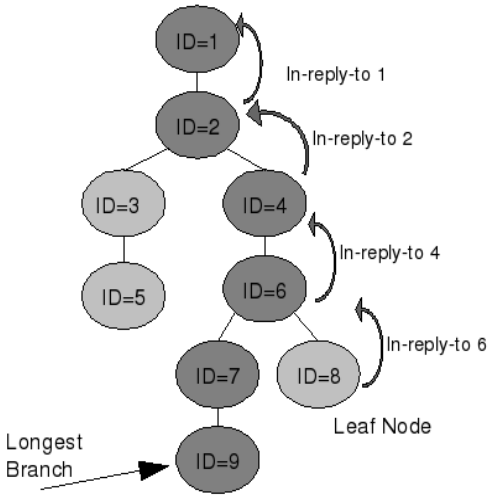

Figure 3.6: Overview of thread structure and terms, showing in-reply-to structure and longest branch.

structure of the thread is a consideration. Threads can be examined a) on a branch-by-branch basis (essentially treating each branch as a distinct thread), b) the branching structure can be ignored, focusing instead on the relative time ordering of the messages, or c) a *representative* branch can be selected, such as the longest branch, to represent the thread as a whole. For our purposes, a) was deemed to time consuming, b) was deemed impractical due to unknowns regarding timezones of intermediate mail servers, as well as the fact that often people do sometimes reply to older messages in a thread on purpose. Thus, the 'representative branch' approach was used.

After threads were reconstructed, they were stored in an internal Thread object and saved to disk in JSON format. A custom JSON-encoder and decoder were used.

3.2.4 Quoted Text

In order to properly analyze conversations, a way is needed to distinguish between original content in e-mail bodies, and the category of text we call 'quoted text.' Quoted text arises from three general categories: reply text is text quoted from one or more previous e-mails, and typically appears either inline, or appended to the current e-mail; signatures, which may be repeated many times in the course of a conversation thread but bear no relevance to the content (similar in some ways to stop words); and forwarded text, that may not have been present in

previous e-mails, and may well be relevant to the conversation at hand, but is not original text from the author of the message.

A simple rule-based approach was applied using regular expressions to remove quoted text and forwarded messages. Each message is parsed for lines beginning with typical quoted-text symbols (one or more of the pipe ('—'), and the greater-than symbol ('>')). Patterns are developed to match lines which mark the beginning of a quoted section, such as the many possible varieties of prefixes similar to "On May 1, 2009, jessy <jessy.cowansharp@gmail.com> wrote:", or lines marking the beginning of a forwarded message, e.g. "——— Forwarded message ———-".

Alternatively, quoted text can be interpreted as key to giving context to a discussion, and repeating what someone has said might legitimately give higher weighting to their words. Further, if someone quotes a paragraph or sentence of a correspondent, and simply says, "yes," then removing that text could be more harmful than helpful in terms of understanding the evolution of the conversation.

We call the versions of the e-mail threads with the quoted text removed *processed* threads, and the version with all the original content *raw* threads. In the e-mail thread classification experiment in Section 4.1, we compare results with both processed and raw threads.

3.2.5 Stop Words

It is typical in many NLP applications to remove what are called stop words from text being analyzed. Stop words have high frequency but low meaning; they are words which stitch sentences together, such as *the*, *it* and *at*. Unfortunately, stop words are language dependent, and must be manually identified. Many corpora also have custom stop words as a function of their topic.

For our applications, the English-language stop list was used from the Python Natural Language Toolkit (NLTK) [1]. In addition, a set of custom stop words were identified through initial analysis and removed as well. The full list of stop words, both generic and custom, is contained in Appendix B.

3.2.6 Tokenization and Stemming

For our purposes in these experiments we use a very simple tokenizer which creates strings from groups of alphanumeric characters. It's acceptable if certain words are tokenized incorrectly or

somewhat arbitrarily, e.g., times (8:00 gets tokenized into 8 and 00, or http://example.com/index gets tokenized into *http*, *example*, *com* and */index*)– as long as the tokenization is consistent. However, it would be undesireable for words with contextual punctuation to be treated as different from ones without– eg. "no" vs. `no`, or `*menu*` vs. `menu`, or `(for me?)` vs `me`.

Two types of word stemming, Porter and Lancaster, were applied to message bodies. The ideal word stymie would replace word tokens with their morphological roots. However, this is a rather difficult task in practice, and so various stemmers have been developed which take slightly different approaches to normalizing different tenses and possible conjugations of tokens.

Character n-grams have been used as an alternative to stemming in some applications; the idea being that a good choice of n will have a similar effect, by truncating tokens in a manner similar to a stemmer. Based on the results in [5], we try character 5-grams.

CHAPTER 4:
Experiments

These experiments apply Latent Dirichlet Allocation (LDA) to the problems of e-mail-thread classification, and automated detection of topic change.

Since the majority of threads are on a single high level topic, in the first experiment we explore how LDA performs on conversational corpora, by classifying e-mail messages with their corresponding thread. Four distance metrics are used and compared as accuracy measures for this experiment.

In the second experiment, we select a number of e-mail threads where the topic did not remain consistent, and attempt to automatically identify the topics using LDA, and the transition points between topics using DBSCAN.

4.1 Message Thread Classification using LDA

A naive guess about which thread an e-mail belongs to would select the longest thread; without any other information, this is the most likely category. Since this experiment has not been done previously, this becomes the baseline against which the e-mail thread classification experiments are compared. The baseline value is the percent of correct classifications we would expect a system using this naive decision scheme to achieve. The number of messages in the raw and processed thread groups are the same, and so the baselines are the same. The baselines for control groups of size 50, 100, 150, 500, and 1000, are given in Table 4.1.

Experiment Baselines for Different Control Group Sizes

Group Size	Longest Branch	Total Messages	Baseline
50	5	250	2%
100	28	594	4.7%
150	28	931	3%
500	36	3489	1.03%
1000	54	7374	0.73%

Table 4.1: Baseline accuracy values for different control groups.

For this experiment, the open source library Machine Learning for Language Toolkit (MALLET) was used [17]). Mallet includes a parametrized interface for the creation of LDA topic

models, a highly efficient implementation of Gibbs sampling, and a number of tools for exploring the algorithm's results.

Using Mallet, a classifier was built, with 20% of the messages from each thread reserved for a test set, and 80% used as training data. The resulting classifier was then applied to classify the test e-mails with their original threads. All experiments were averaged over 5 runs; this number was determined by running one experiment 5, 10, and 15 times, and comparing the variation of the results. Since the variation was within $+/-2$ each time, an average of 5 runs was deemed to be sufficient. The LDA and Gibbs parameters to this model are outlined in Table 4.3 and 4.2, respectively. These values were chosen based on experimental best-practice determined by Steyvers and Griffiths in [19].

LDA Model Parameters

α	50
β	0.01

Table 4.2: LDA Parameters for LDA model of e-mail threads

Gibbs Parameters

Iterations	40
Thinning	3000

Table 4.3: Gibbs Parameters for LDA model of e-mail threads

LDA Topics: Top Words

space	house	nasa	yuri	volunteer
earth	room	gov	night	people
moon	people	ames	space	events
mars	place	arc	event	volunteers
human	craigslist	center	nasa	room
http	living	colab	art	setup
nuclear	mansions	research	science	stage
nations	home	605	www	1
climate	rainbow	http	ames	area
science	move	604	worldspaceparty	table

Table 4.4: Top ten words for 5 of the 50 topics in one of the LDA models, ordered by decreasing weight from the top. Note that these words are representative, and in practice change slightly for each run due to Gibbs sampling. This example also includes stop words which were subsequently removed.

First, the topic weights or probabilities for each thread were calculated. For LDA, this is called estimating, and the result of an estimation is a vector with as many dimensions as there are

topics in the model. Each dimension corresponds to the estimated probability that the words in the e-mail were drawn from that specific topic.

In order to classify e-mails, these topic probability weight vectors were compared with that of each thread, and the e-mail was classified as coming from the thread that was most similar.

Four metrics were used to measure similarity, in order to compare their results: cosine similarity, euclidean distance, the entropy-based measure KL-divergence, and the symmetric version of KL-divergence, Shannon divergence.

To begin with, we started with 100 threads, and results were obtained using Porter stemming, Lancaster stemming, no stemming, and character 5-grams. In Tables 4.5 and 4.6, we compare the results of these four tokenizing approaches using all four distance metrics.

% Correct over 100 Processed Threads using 50 Topics

Porter	Correct	Lancaster	Correct	No Stemming	Correct	5-gram	Correct
Cosine	34	Cosine	33	Cosine	36	Cosine	16
Euclidean	29	Euclidean	28	Euclidean	26	Euclidean	11
KL	33	KL	30	KL	30	KL	10
Shannon	37	Shannon	35	Shannon	38	Shannon	14

Table 4.5: Results from LDA classification experiment for each of Cosine, Euclidean, and Shannon and KL-divergence distance measures for processed threads. Since the results are a function of sampling, this is an average over 5 runs. Numbers may not sum to 100 due to rounding.

% Correct over 100 Raw Threads using LDA Topics = 50

Porter	Correct	Lancaster	Correct	No Stemming	Correct	5-gram	Correct
Cosine	84	Cosine	84	Cosine	83	Cosine	80
Euclidean	88	Euclidean	85	Euclidean	83	Euclidean	66
KL	86	KL	87	KL	87	KL	77
Shannon	88	Shannon	85	Shannon	84	Shannon	82

Table 4.6: Results from LDA classification experiment for each of Cosine, Euclidean, and Shannon and KL-divergence distance measures *for raw threads*. Since the results are a function of sampling, this is an average over 5 runs. Numbers may not sum to 100 due to rounding.

As can be seen from Table 4.5, the correctly classified e-mails were in the low to high 30% range in most cases, except the character n-grams, which performed worse at 12%. Compared to our baseline of 4.7%, these classification results do show improved performance.

The same experiment is performed with raw e-mail threads, without any quoted reply text, forwarded content, or signatures removed. Table 4.6 clearly shows that having this extra text

to train on is a substantial advantage.

Based on the results of these initial experiments, we can see that Porter very slightly outperforms the other methods for the processed threads, but performs equally well with no stemming at all for the raw threads. Cosine similarity and Shannon distance were also slightly better performers in terms of the distance metric, but not significantly. We continue to calculate the different distance metrics throughout the other experiments.

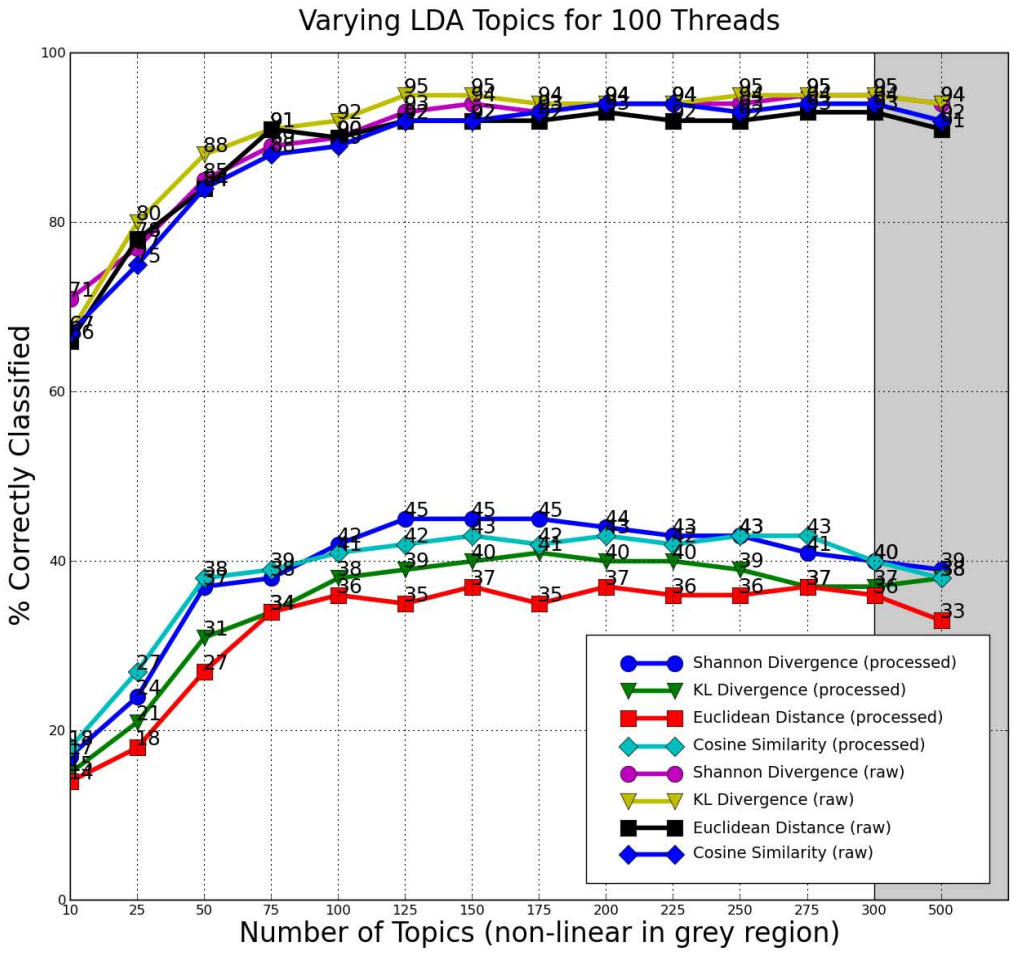

Figure 4.1: E-Mail classification performance for LDA models built with different numbers of topics, for both raw and processed threads. Each line shows a different distance metric. It can be seen that varying the LDA topics parameter improves the results.

Next, the number of topics used as input to the LDA models was varied, to see if these first

results could be improved upon. Porter stemming was used exclusively for this experiment. In Figure 4.1, we can see that by varying the number of topics, the results for the processed threads (the bottom group) go as high as 45% for the Shannon divergence, with the optimal number of LDA topics at 150 or 175. The performance for LDA topics less than 50 is significantly worse than for the higher values. Other metrics follow a similar curve, with Cosine and Shannon competing (marginally) for first place.

For the raw threads, as seen in the top group of Figure 4.1, there is a similar sharp improvement in accuracy up to 75 LDA topics. The best results are obtained at a value of 150 or higher. It looks like the results may be starting to dip down again after 200 LDA topics, but results for the Euclidean metric stay roughly the same. This graph would need to be extended to higher LDA topic numbers to verify if the results continue to decrease.

For processed threads, the optimal number of topics seemed to be 150 LDA topics, while for raw threads we selected 275. Holding the topic number constant at these values for processed and raw threads, respectively, the number of threads is increased to see how the model will perform on larger control groups. Figure 4.2 shows how LDA performs when run on 50, 100, 150, 500, and 1000 threads. As in the previous experiments, a 20/80 split was used, building a model from 80% of the data, and testing it on the remaining 20%.

The accuracy of the classification results decreases as the number of threads goes up, for both processed and raw threads. Although increasing the number of threads increases the amount of data to train on, it also increases the choices the classifier has when selecting a thread to associate a test e-mail with. This suggests that the noise in the data is increasing faster than the quality of the topic models, resulting in decreased accuracy. Given the corpus, this is not terribly suprising. We explore this further in Chapter 5.

The results for the raw threads are quite good here, even up to 1000 threads, while the results for the processed threads is fair up to about 150 threads, then dips under 25% at 500 threads, and achieves between 13 and 15% accuracy for 1000 threads. The results are still better than the baseline of 0.73%.

For the 1000 thread control groups, the LDA topic variation experiment was re-run to see if the results differed for the larger number of threads. Figure 4.3 shows the results for the processed and raw threads. The processed threads are the buttons group, and the raw threads are the top group. The processed threads experience a minor improvement in classification

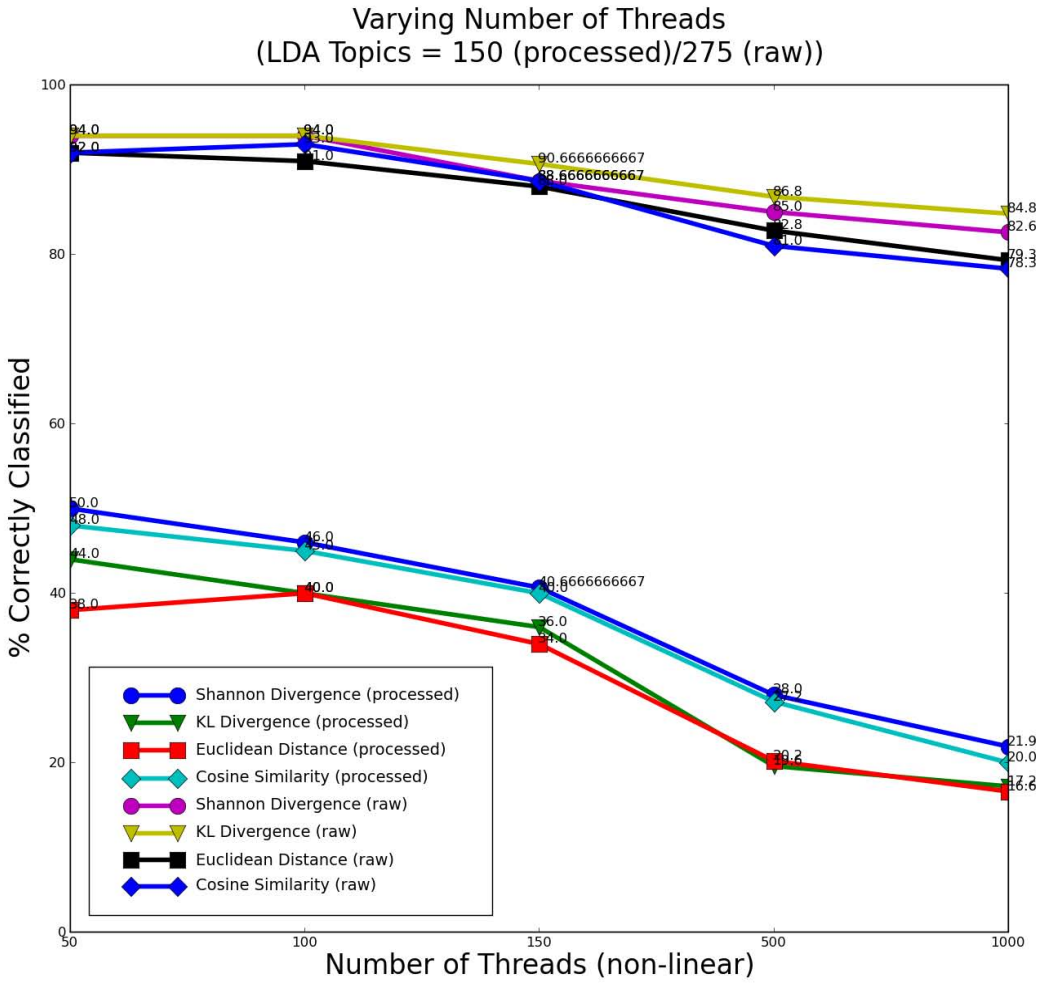

Figure 4.2: A Comparison of e-mail classification performance for LDA models built with increasing numbers of threads. Results are shown for both processed threads (lower group) and raw threads (upper group). Each line shows a different distance metric. Performance decreases as the number of threads goes up.

results as the number of LDA topics is increased up to 500, which is interesting since for the 100 thread group, this number of topics resulted in decreased performance. The raw threads also demonstrate increased performance, but with a lower overall gain.

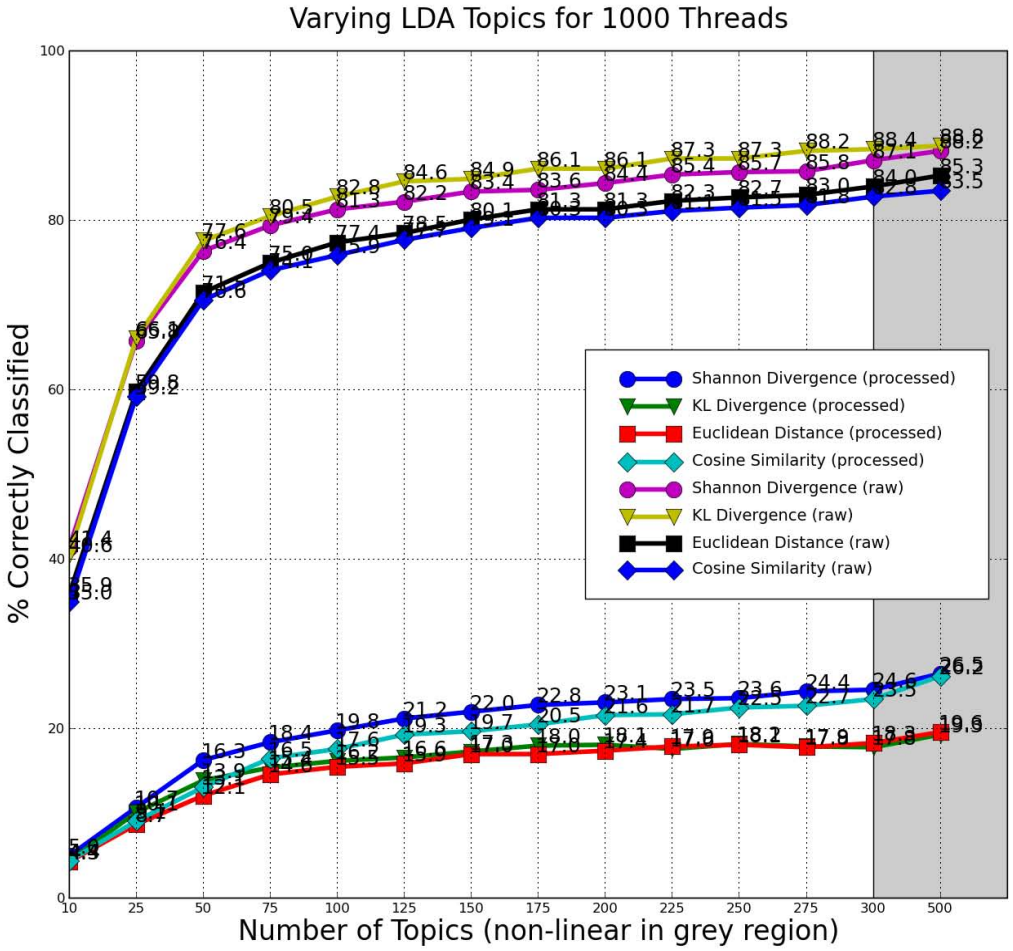

Figure 4.3: Performance as the number of LDA topics is increased, for 1000 processed threads.

4.2 Automated Detection of Topic Change

While the majority of e-mail threads are on a single topic, a conversation can switch focus within a thread, either completely, or by moving into a different but related area. The question arises whether we can accurately define, detect, and measure such changes.

Two approaches to detecting topic change were taken. Both were based on topic models built using Mallet's implementation of Latent Dirichlet Allocation. All 2168 e-mails in the corpus were used to train a model, from which the weight of each word for each topic was determined.

As can be seen in Figure 4.4, there is an exponential falloff in the weight or probability of each word for a given topic. Because of the probabilistic nature of LDA, and the smoothing applied, every word has at least some minimal probability of having being drawn from each topic. Thus, for the first approach, the top N keywords from each topic in the topic model were extracted. A clustering algorithm was applied to the resulting output to identify subtopics, and transition points identified corresponding to topic changes.

For the second approach, a sliding window technique was used. Each window was classified as belonging to a specific topic by calculating the total weight of the words in one window for each topic, and selecting the topic with the maximum value. As the window moved across the thread, if the topic classification of the window differed from the previous one, a topic transition was identified.

The keyword clustering experiment involved iterating over the words of a thread; if a keyword appeared, this was taken as an indicator of the topic. Simplistically, as different keywords appear over the thread, if they belong to a different topic, then a topic change is considered to have occured.

One of the challenges with this approach is that certain terms, like 'http', and 'NASA' are in the top N words for many topics. In certain cases, this may be a function of the relatively small sample size, but it is also the case that many individuals will have cross-cutting themes in their personal communications.

In addition, certain words are genuinely content, but have a high frequency in the corpus and thus in multiple topics, because of natural commonalities in the topics of an individual's e-mails. Such frequent keywords have the same problem as stop words–they dilute the topic assessment because they do not provide a clear indication of topic.

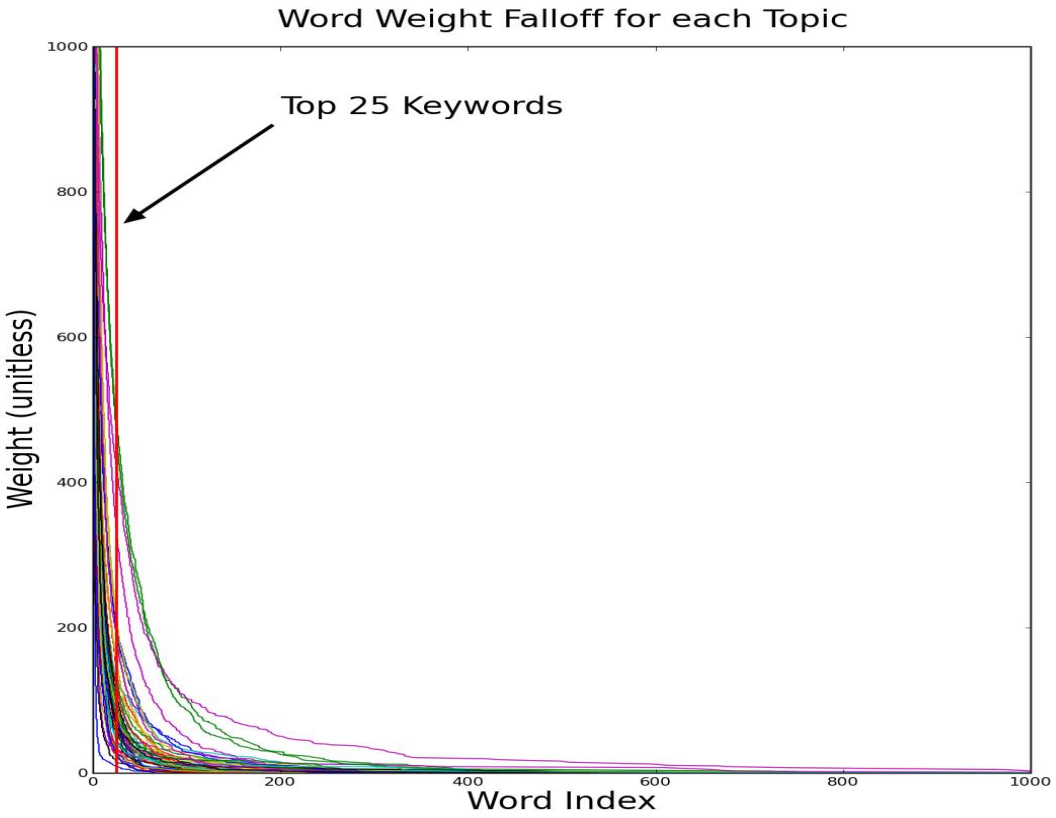

Figure 4.4: This figure shows the keywords per topic sorted in order of decreasing weight (or relevance) to that topic. The x-axis shows the word ordering, while the y-axis shows the weight of a word for a given topic. There is one line per topic. The words themselves are not displayed; rather, the chart is shown to emphasize the exponential falloff of word-topic weights.

A threshold, T, is chosen for the number of topics a word can be a keyword for before it becomes too diluted. If a word exceeds the threshold, it is discarded from the keyword list, and the next most frequent word in that topic (that is not also too common) replaces it. What we end up with is a list of key *distinguishing* words. Others have taken a more formal entropy-based approach [21], but that was not explored here. Higher values of N imply a more relaxed topic definition, accounting for more peripheral words in a topic; thus, we can think of N as a relaxation factor.

If T is increased, a keyword can be present for more topics, meaning that they will be less unique. In a way, this can be thought of as playing as similar role to the hyperparameter α in the Dirichlet distribution. Increasing T increases the amount of keywords a topic can share, or how much they overlap.

Keyword Selection Parameters

Parameter	**Effect**
N (Keywords)	Relaxation
T (Threshold)	Uniqueness

Table 4.7: High level effects of varying the parameters N and T in the selection of keywords used in topic identification.

If a low threshold is chosen, the resulting output will emphasize words that were more indicative of that topic. If N is also decreased, the distinguishing words are those which are increasingly related to that topic. In practice, the result is a squeezing effect that emphasizes groups of words which appear frequently in the same grouping, such as newsgroup footers or frequent correspondents' signatures.

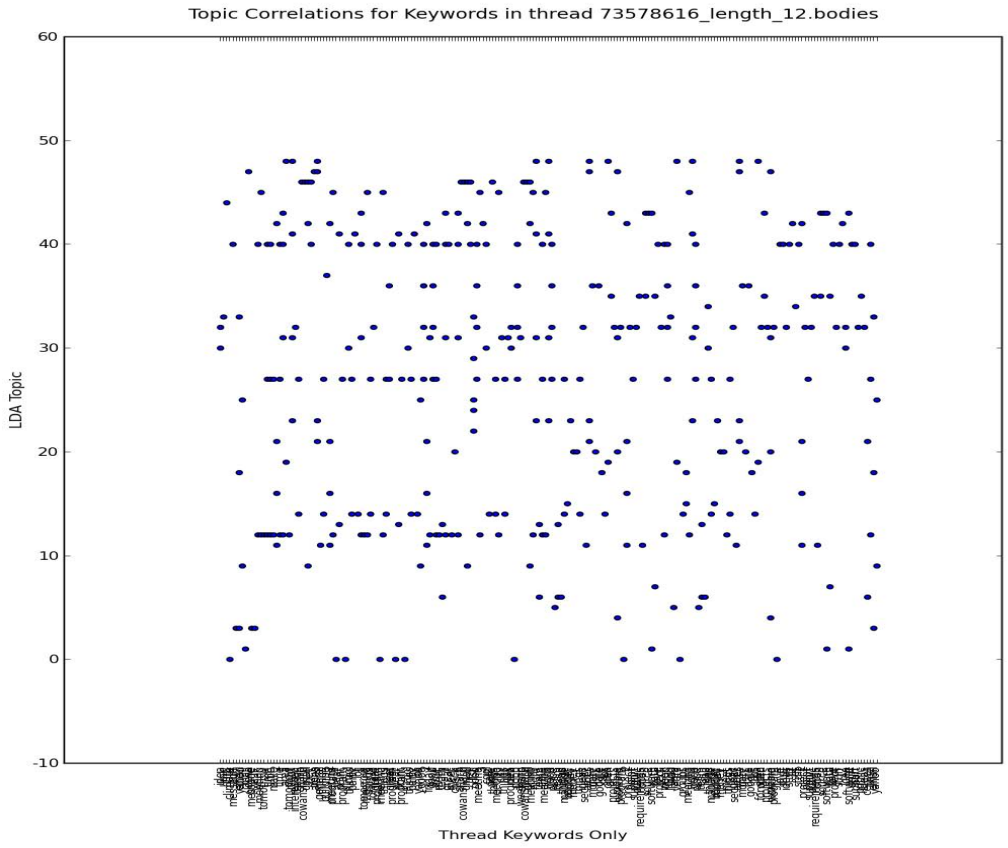

Figure 4.5: Topic correlations for keywords in a single e-mail thread. LDA Topics = 50; Keywords = 10; Threshold = 5. We can see that structure emerges in the thread. Note that the x-axis shows words in the order they appeared in the thread, and thus also correlates with time.

In Figure 4.5 it can be seen that this method begins to indicate some structure in the threads. In order to automate detection of significant changes, a custering algorithm is applied to the data. The goal of the clustering is to determine where the groupings of points in Figure 4.5 are significant, and how to determine the boundaries of topics.

The clustering algorithm used is called DBSCAN, as described in Section 3.1.6. DBSCAN is a density based clustering algorithm, with the important characteristic that it discovers, instead of taking as input, the number of clusters in a given set of data points. DBSCAN has two input parameters, ϵ, and $min\_pet's$. ϵ defines the radius around a given point that is searched for neighbours, while min_pts defines the minimum number of neighbours that radius must contain.

For this application of the DBSCAN algorithm, we only want points to be clustered with other points from the same topic. Since a point can only be considered a neighbour of another point if it's within a distance of ϵ or less, to enforce clustering within topics only, the feature vectors passed to DBSCAN are defined such that each topic is its own dimension, and the distance between topics is always greater than ϵ.

Larger ϵ means fewer, but larger, clusters. Similarly, as min_pts is increased to values closer to ϵ, the algorithm will find more densely connected regions. In this experiment, because we are only clustering within topics, min_pts represents the number of keywords in a given window that should be part of the same LDA topic, before we consider it representative of a topic of the thread. In other words, min_pts is the minimum number of points needed to form a coherent topic within the thread. ϵ is a measure of how tightly bound the topics are. If the ratio of $\min\_pts$ to ϵ stayed the same, but both values grew larger, it would be as though the focus of the clustering had blurred, or the edges of each group were less well-defined.

We want to choose a min_pts/ϵ ratio that will be sensitive enough to detect changes in topic, but forgiving enough to account for the uncertainty introduced by the probabilistic nature of word-topic associations in LDA.

Finally, in order to mark the transitions between the identified clusters (topics), the mid-point between the end of one cluster and the beginning of another was selected. Topic changes at the first word of a thread were discarded.

Twelve threads were identified that contained one or more clear topic changes. These threads typically either had a main topic, with a detour in the middle from which it turned back on track,

or, exhibited a complete topic change from which it did not return. The former threads contained two changes, one to change topic onto the detour topic, and the second to change topics back. These detour topics are typically shorter than the parent topic. The latter threads exhibited only a single change. The second topics were often items that the original topic reminded the author to bring up, while others were unrelated to any contextual information.

Between the keyword selection and the DBSCAN parameters, 4 variable values must be selected for each run of the experiment. In Tables 4.8 to 4.13, the results are shown for 6 runs of the experiment over 12 threads, using the word-topic weights from LDA with the number of topics set to 50. The parameter inputs for keyword selection and the DBSCAN algorithm are included in the table header as a 4-tuple representing (N, T, ϵ, min_pts). The number and location of topic changes identified were tracked, and compared to the number of actual changes in the thread (determined by hand labelling), and their correct locations. Precision and Recall were calculated for all threads. The location accuracy was determined to within $+/-$ one sentence.

Figure 4.6 shows the results of clustering and transition point identification for a single thread. The top plot shows the LDA topics plotted for the selected keywords and threshold, and the bottom shows the clusters after scanning. The vertical lines are automatically determined transition points. The x-axis shows the keywords in the same order they appeared in the original thread, and thus also represent the time dimension. Note that in this (and subsequent) images, only the top N keywords are shown. Points which were not allocated to any cluster are considered noise, and they are shown in a light gray colour on the bottom plot.

The second approach taken was the sliding window-based technique. Recall that each word has a certain weight associated with each topic. In general, words have a high weight in association with only a few topics. Thus, for each window of size N words, the corresponding word weights for each topic were summed, and the topic with highest associated value was selected. A transition was identified as occuring at the middle of the window ($N/2$) location. As the window was moved across the text of the thread, the topic could be seen to change at specific points, as seen in Figure 5.3.

The challenge with the window classification scheme was to select a window size large enough to smooth over elements like signatures, but small enough to capture genuine topic changes. Trial and error resulted in the selection of 20, 40 and 100 for the window sizes. The results for the sliding window experiment applied to the 12 threads are shown in Tables 4.14 to 4.16.

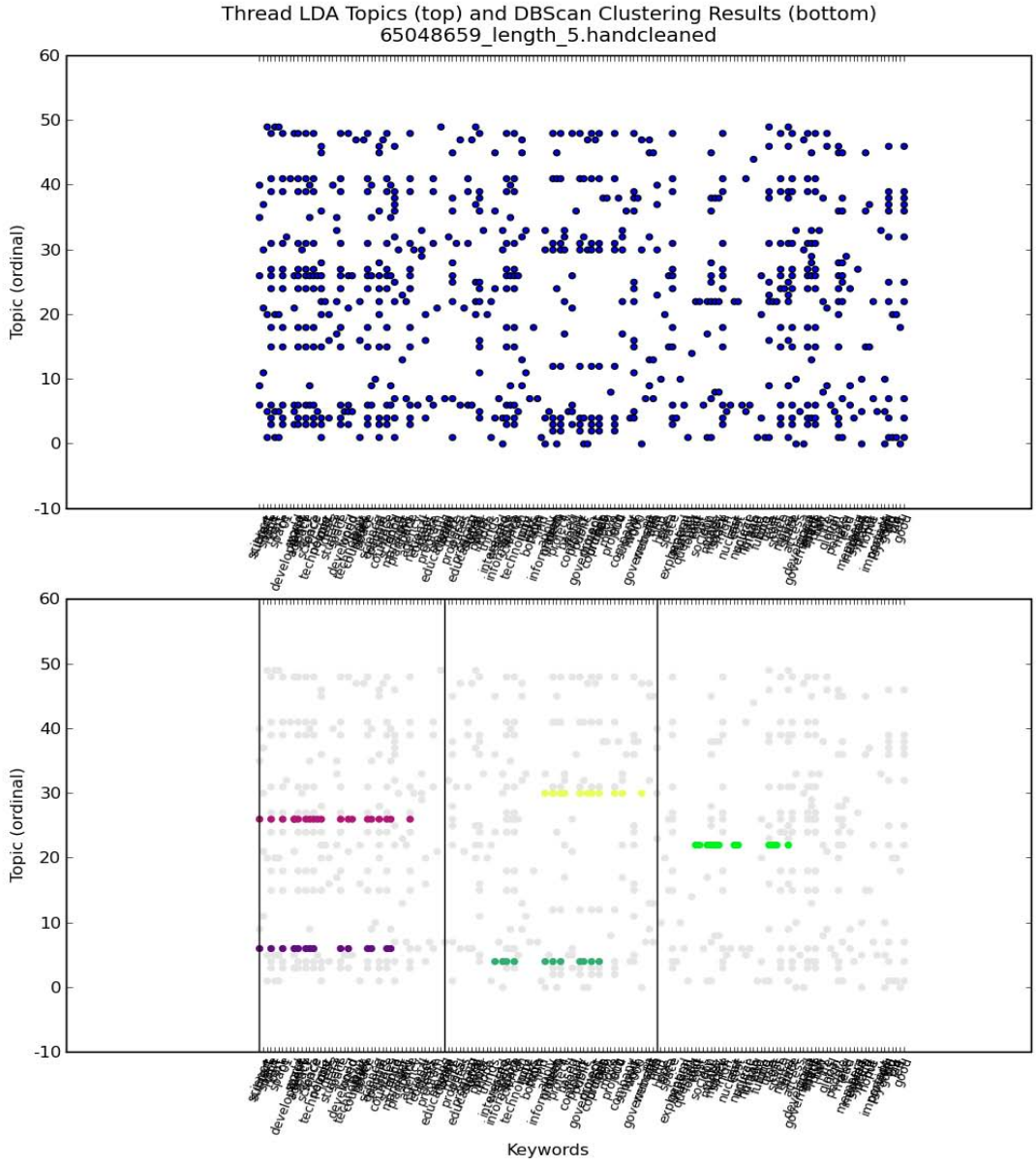

Figure 4.6: An example of the clustering results for a single thread. Threshold T=25, N=10 $min\_pts = 4$ and $\epsilon = 6$ (LDA Topics = 50). Vertical lines show identified topic transitions. Personally identifying terms are grayed out for privacy purposes.

DBSCAN - (N=25, T=5, E=6, MP=4)								
Thread ID	Len (ch)	Msgs	# Changes	# Found	Cor. Loc	P	R	F
75273856_length_8	2097	8	1	1	0	0	0	0
53773016_length_7	908	7	1	1	0	0	0	0
78378912_length_15	4530	14	1	4	1	0.25	1	0.4
85425331_length_6	1820	6	2	1	0	0	0	0
83944656_length_9	3450	9	1	1	1	1	1	1
83872656_length_15	4148	15	3	1	0	0	0	0
59502032_length_13	2714	13	2	1	1	1	0.5	0.67
75215072_length_7	3868	7	1	2	0	0	0	0
65048659_length_5	14436	5	1	3	0	0	0	0
83864104_length_18	8743	18	1	5	1	0.2	1	0.33
73555624_length_7	6063	7	1	2	0	0	0	0
53638713_length_6	1306	6	1	1	1	1	1	1

Table 4.8: DBSCAN results for (N=25, T=5, E=6, MP=4). Results show the Thread ID, thread length in characters (len (ch)), number of messages in thread branch, the number of actual topic changes in the thread (# Changes), the number of topic changes found (# Found), and the number found in the correct location (Cor. Loc). P, R, and F give the precision, recall and F-score, as defined in 3.1.7

DBSCAN - (N=25, T=5, E=10, MP=6)								
Thread ID	Len (ch)	Msgs	# Changes	# Found	Cor. Loc	P	R	F
75273856_length_8	2097	8	1	0	0	0	0	0
53773016_length_7	908	7	1	0	0	0	0	0
78378912_length_15	4530	14	1	1	0	0	0	0
85425331_length_6	1820	6	2	0	0	0	0	0
83944656_length_9	3450	9	1	1	1	1	1	1
83872656_length_15	4148	15	3	1	0	0	0	0
59502032_length_13	2714	13	2	0	0	0	0	0
75215072_length_7	3868	7	1	1	0	0	0	0
65048659_length_5	14436	5	1	1	0	0	0	0
83864104_length_18	8743	18	1	3	0	0	0	0
73555624_length_7	6063	7	1	3	0	0	0	0
53638713_length_6	1306	6	1	1	1	1	1	1

Table 4.9: (DBSCAN results for N=25, T=5, E=10, MP=6). Results show the Thread ID, thread length in characters (len (ch)), number of messages in thread branch, the number of actual topic changes in the thread (# Changes), the number of topic changes found (# Found), and the number found in the correct location (Cor. Loc). P, R, and F give the precision, recall and F-score, as defined in 3.1.7

DBSCAN - (N=10, T=5, E=6, MP=4)								
Thread ID	Len (ch)	Msgs	# Changes	# Found	Cor. Loc	P	R	F
75273856_length_8	2097	8	1	0	0	0	0	0
53773016_length_7	908	7	1	0	0	0	0	0
78378912_length_15	4530	14	1	3	0	0	0	0
85425331_length_6	1820	6	2	0	0	0	0	0
83944656_length_9	3450	9	1	1	0	0	0	0
83872656_length_15	4148	15	3	3	2	0.67	1	0.8
59502032_length_13	2714	13	2	1	1	1	0.5	0.67
75215072_length_7	3868	7	1	0	0	0	0	0
65048659_length_5	14436	5	1	1	0	0	0	0
83864104_length_18	8743	18	1	4	1	0.25	1	0.4
73555624_length_7	6063	7	1	4	1	0.25	1	0.4
53638713_length_6	1306	6	1	1	1	1	1	1

Table 4.10: DBSCAN results for (N=10, T=5, E=6, MP=4). Results show the Thread ID, thread length in characters (len (ch)), number of messages in thread branch, the number of actual topic changes in the thread (# Changes), the number of topic changes found (# Found), and the number found in the correct location (Cor. Loc). P, R, and F give the precision, recall and F-score, as defined in 3.1.7

DBSCAN - (N=10, T=5, E=10, MP=6)								
Thread ID	Len (ch)	Msgs	# Changes	# Found	Cor. Loc	P	R	F
75273856_length_8	2097	8	1	0	0	0	0	0
53773016_length_7	908	7	1	0	0	0	0	0
78378912_length_15	4530	14	1	1	0	0	0	0
85425331_length_6	1820	6	2	0	0	0	0	0
83944656_length_9	3450	9	1	1	0	0	0	0
83872656_length_15	4148	15	3	3	3	1	1	1
59502032_length_13	2714	13	2	1	0	0	0	0
75215072_length_7	3868	7	1	0	0	0	0	0
65048659_length_5	14436	5	1	1	0	0	0	0
83864104_length_18	8743	18	1	1	0	0	0	0
73555624_length_7	6063	7	1	2	0	0	0	0
53638713_length_6	1306	6	1	1	1	1	1	1

Table 4.11: DBSCAN results for (N=10, T=5, E=10, MP=6). Results show the Thread ID, thread length in characters (len (ch)), number of messages in thread branch, the number of actual topic changes in the thread (# Changes), the number of topic changes found (# Found), and the number found in the correct location (Cor. Loc). P, R, and F give the precision, recall and F-score, as defined in 3.1.7

DBSCAN - (N=25, T=51, E=15, MP=10)								
Thread ID	Len (ch)	Msgs	# Changes	# Found	Cor. Loc	P	R	F
75273856_length_8	2097	8	1	0	0	0	0	0
53773016_length_7	908	7	1	0	0	0	0	0
78378912_length_15	4530	14	1	1	0	0	0	0
85425331_length_6	1820	6	2	0	0	0	0	0
83944656_length_9	3450	9	1	1	0	0	0	0
83872656_length_15	4148	15	3	1	0	0	0	0
59502032_length_13	2714	13	2	0	0	0	0	0
75215072_length_7	3868	7	1	1	0	0	0	0
65048659_length_5	14436	5	1	2	0	0	0	0
83864104_length_18	8743	18	1	1	0	0	0	0
73555624_length_7	6063	7	1	3	0	0	0	0
53638713_length_6	1306	6	1	1	1	1	1	1

Table 4.12: DBSCAN results for (N=25, T=51, E=15, MP=10). Results show the Thread ID, thread length in characters (len (ch)), number of messages in thread branch, the number of actual topic changes in the thread (# Changes), the number of topic changes found (# Found), and the number found in the correct location (Cor. Loc). P, R, and F give the precision, recall and F-score, as defined in 3.1.7

DBSCAN - (N=25, T=51, E=5, MP=3)								
Thread ID	Len (ch)	Msgs	# Changes	# Found	Cor. Loc	P	R	F
75273856_length_8	2097	8	1	0	0	0	0	0
53773016_length_7	908	7	1	1	0	0	0	0
78378912_length_15	4530	14	1	7	1	0.14	1	0.25
85425331_length_6	1820	6	2	1	0	0	0	0
83944656_length_9	3450	9	1	3	0	0	0	0
83872656_length_15	4148	15	3	7	3	0.43	1	0.6
59502032_length_13	2714	13	2	1	0	0	0	0
75215072_length_7	3868	7	1	5	1	0.2	1	0.33
65048659_length_5	14436	5	1	7	1	0.14	1	0.25
83864104_length_18	8743	18	1	6	1	0.17	1	0.29
73555624_length_7	6063	7	1	7	1	0.14	1	0.25
53638713_length_6	1306	6	1	1	1	1	1	1

Table 4.13: DBSCAN results for (N=25, T=51, E=5, MP=3). Results show the Thread ID, thread length in characters (len (ch)), number of messages in thread branch, the number of actual topic changes in the thread (# Changes), the number of topic changes found (# Found), and the number found in the correct location (Cor. Loc). P, R, and F give the precision, recall and F-score, as defined in 3.1.7

Window Size = 20								
Thread ID	Len (ch)	Msgs	# Changes	# Found	Cor. Loc	P	R	F
75273856_length_8	2097	8	1	4	0	0	0	0
53773016_length_7	908	7	1	2	1	0.5	1	0.67
78378912_length_15	4530	14	1	23	0	0	0	0
85425331_length_6	1820	6	2	14	1	0.07	1	0.13
83944656_length_9	3450	9	1	10	1	0.1	1	0.18
83872656_length_15	4148	15	3	8	1	0.13	1	0.22
59502032_length_13	2714	13	2	11	2	0.18	1	0.31
75215072_length_7	3868	7	1	27	0	0	0	0
65048659_length_5	14436	5	1	76	1	0.01	1	0.03
83864104_length_18	8743	18	1	48	1	0.02	1	0.04
73555624_length_7	6063	7	1	24	1	0.04	1	0.08
53638713_length_6	1306	6	1	1	1	1	1	1

Table 4.14: Sliding window results for window size = 20. Results show the Thread ID, thread length in characters (len (ch)), number of messages in thread branch, the number of actual topic changes in the thread (# Changes), the number of topic changes found (# Found), and the number found in the correct location (Cor. Loc). P, R, and F give the precision, recall and F-score, as defined in 3.1.7

Window Size = 40								
Thread ID	Len (ch)	Msgs	# Changes	# Found	Cor. Loc	P	R	F
75273856_length_8	2097	8	1	2	0	0	0	0
53773016_length_7	908	7	1	0	0	0	0	0
78378912_length_15	4530	14	1	8	0	0	0	0
85425331_length_6	1820	6	2	3	0	0	0	0
83944656_length_9	3450	9	1	7	1	0.14	1	0.25
83872656_length_15	4148	15	3	4	1	0.25	1	0.4
59502032_length_13	2714	13	2	6	1	0.17	1	0.29
75215072_length_7	3868	7	1	4	0	0	0	0
65048659_length_5	14436	5	1	54	1	0.02	1	0.04
83864104_length_18	8743	18	1	48	1	0.02	1	0.04
73555624_length_7	6063	7	1	7	0	0	0	0
53638713_length_6	1306	6	1	0	0	0	0	0

Table 4.15: Sliding window results for window size = 40. Results show the Thread ID, thread length in characters (len (ch)), number of messages in thread branch, the number of actual topic changes in the thread (# Changes), the number of topic changes found (# Found), and the number found in the correct location (Cor. Loc). P, R, and F give the precision, recall and F-score, as defined in 3.1.7

Window Size = 100								
Thread ID	Len (ch)	Msgs	# Changes	# Found	Cor. Loc	P	R	F
75273856_length_8	2097	8	1	0	0	0	0	0
53773016_length_7	908	7	1	0	0	0	0	0
78378912_length_15	4530	14	1	2	0	0	0	0
85425331_length_6	1820	6	2	0	0	0	0	0
83944656_length_9	3450	9	1	1	0	0	0	0
83872656_length_15	4148	15	3	0	0	0	0	0
59502032_length_13	2714	13	2	0	0	0	0	0
75215072_length_7	3868	7	1	4	0	0	0	0
65048659_length_5	14436	5	1	21	0	0	0	0
83864104_length_18	8743	18	1	28	1	0.04	1	0.07
73555624_length_7	6063	7	1	1	1	1	1	1
53638713_length_6	1306	6	1	0	0	0	0	0

Table 4.16: Sliding window results for window size = 100. Results show the Thread ID, thread length in characters (len (ch)), number of messages in thread branch, the number of actual topic changes in the thread (# Changes), the number of topic changes found (# Found), and the number found in the correct location (Cor. Loc). P, R, and F give the precision, recall and F-score, as defined in 3.1.7

CHAPTER 5:
Analysis and Future Work

In this chapter we discuss the results for both thread classification and the topic change detection experiments. The raw versus processed versions of the data set are considered, and the relative performance of the four distance metrics is compared. Characteristics and challenges of the data set are identified, and the definition and implementations of topic and topic change are explored.

The conversational aspect of e-mail threads gives rise to high variance in vocabulary usage over the thread space, with a large number of contextual terms and references to concepts, experiences or discussions external to the thread itself. There are less nouns and other anchoring terms to train on, and more implicit assumptions about common ground and shared knowledge. In addition, e-mail communications are often short and informal, reducing meaningful context even futher.

All these factors make e-mail conversations a rather challenging data set to analyze, and this is borne out in the results from Chapter 4 and discussed further below. At the same time, these are new areas of research, and it is our hope that baselining performance with these techniques will help to identify directions where future research can lead to improvements.

5.1 E-Mail Thread Classification

For the thread classification experiments, overall we saw decreasing performance for larger numbers of threads. Performance was substantially increased for classification of raw threads versus processed threads, due to the additional context.

The best results were around 35% for the processed threads, and just over 90% for the raw threads. Porter stemming, Lancaster stemming, and no stemming performed roughly the same, with character 5-grams performing distinctly worse in both cases. The Cosine and Shannon distance metrics performed slightly better on the processed threads, and the KL-divergence performed marginally better for the raw threads, but not significantly.

Stemming is a difficult task, so one explanation for the lack of improvement is simply that the stemming algorithms performed poorly on the text they were given. An inspection of the

Porter and Lancaster stemmed words does reveal that many words were stemmed incorrectly. For example, these words were stemmed as follows, using the Porter and Lancaster stemmers, respectively:

manage legal financial → Porter Stemmer → manag legal financi
manage legal financial → Lancaster Stemmer → man leg fin

An incorrect word wouldn't *reduce* the accuracy of the results, since all instances of that word would be stemmed in the same way, and therefore their counts in the LDA topic models would be unchanged. The advantage of successful stemming is rather that 'managing' and 'manage' will get counted as the same word. Incorrect stemming simply means that the words counted will be 'manage' and 'manag' (for example). Another possible explanation is that our results are dominated by the LDA topic model quality, and slight changes in token counts may not matter much if the underlying topic model is not strong.

It is interesting that the character n-grams performed substantially worse than the other techniques. Because n-grams are implemented as a sliding window, many are made up of the end of one word and the beginning of another word, implicitly giving them more context. In contrast, the other stemming techniques all use whitespace for token boundaries, and thus do not include context. That the techniques without this context performed better, might mean that words are not being frequently repeated in the same sequences. This could be a by-product of the fact that an e-mail corpus has many authors, thereby dominating any effects that an individual author's 'voice' might have. It could also be a function of swiftly changing contexts and tenses, or the casual nature of the medium.

Varying the number of LDA topics did have a measurable impact on the performance of the classification task. In all cases, there was a dramatic drop in performance for very low values of the LDA topics parameter. Considering the size and nature of the data set, it could legitimately be the case that such a low number of topics is simply a poor fit for the data, causing multiple 'real' topics in the data to be conflated. Additionally, because cosine similarity measures angular distance, it is much less sensitive to differences in values within dimensions than across them. Thus in general we would expect the cosine difference measure to perform slightly worse at lower topic values.

We did see that, for all the topic variation experiments for both the 100- and 1000-thread control groups, the divergence in performance between the different distance metrics was lower when

the results were either very good or very bad. That is, the extreme good or bad results seemed to be more agnostic to the distance metric. Perhaps the differences in the categories (topics) are greater than the variation in the distance measures, as the quality of the model goes up (or down). Using the convergence of multiple distance measures might be an interesting technique for assessing the quality of the models in future experiments.

As the number of threads was increased, a corresponding decrease in the accuracy of the thread classification was also observed. At first glance this isn't suprising; more threads to choose from means, all else being equal, that there is a lower probability of selecting the correct thread for classification. However, additional threads also mean significantly more training data, and thus one would hope for it to increase the quality of the topic models, enough to maintain or improve the classification results. Instead, what we see is that the noise factor increases more quickly than the quality of the topic models.

If additional threads are not refining existing topics, then it may instead be adding new topics, suggesting that the corpus, and thus the topics of this individual's e-mails, contain more breadth than depth overall.

The optimal number of topics for the LDA model changed for the larger thread groups. This is implies that 100 threads was not a large enough subset to be representative of the broader set of topics. As would be expected, the number of LDA topics increased for the larger control group, indicating that there were more latent topics.

In general, there were several factors affecting the quality of the results for processed threads independent of the algorithm applied. Fully cleaning an e-mail corpus from scratch is a formidable task, and our desire to work with threads, as opposed to individual e-mails, made it infeasible to use the more common Enron corpus (because the required header fields were not included). Many Usenet and modern newsgroups have been archived and are stored online for research or analysis purposes, but none that we came across had been pre-processed. There was no immediately clear advantage to using these newsgroups over a corpus belonging to the author, while using the author's corpus did provide the bonus of familiarity with the contents. However, in retrospect, newsgroups may have offered a more rich conversational medium, with longer messages and more substantially fleshed out topics.

Due to the private nature of an individual's e-mail corpus, this makes the data set difficult to sanitize for public release, and thus rigorous verification of research results obtained using it

not possible. These are strong arguments against using personal data sets. While newsgroups would address this because many are public, it would also be useful if the community pooled efforts to clean and release individual e-mail corpora.

The major effort in cleaning such a data set involves the removal of extraneous content; in particular, quoted reply text, forwarded content, and signatures. Hand removal was impractical for the timeframe and resources at hand, and so a best-effort attempt was made to automate the cleaning process (see Section 3.2). Unfortunately, the result was that many replies were missed, reducing the quality of the data set and making it more difficult to assess all the factors influencing the results. In particular, since the results with raw threads were so high, it is possible that stray reply content may have artificially inflated some of the results for processed threads.

Signatures were an equally difficult challenge. They were not removed, since there was no clear way to do this in an automated fashion. Although they represented a smaller proportion of the text than quoted replies and forwards, they are not actually content per se. On one hand, an e-mail thread between two authors would legitimately recognize those authors' respective e-mail signatures as being indicators of membership in a thread. Further, depending on the task, the raw (un-processed) thread content might be the only data on hand (for example, file carving in digital forensics). On the other hand, from a topic modeling standpoint, it's a bit dishonest, since it is not genuinely topical. Worse, for shorter threads, signature text might even dominate the message bodies.

As a function of the casual nature of e-mail, there are likely to be more spelling errors in e-mail conversations than published documents, causing mis-spelled words to be counted as distinct tokens. A simple improvement would probably be to run a spell-check over the message bodies before other processing was conducted.

For the raw threads, given the high frequency with which prior messages were quoted in replies, running the classifier on the raw threads was equivalent to training on the test data. Although the raw thread results provide an upper bound, we're really interested in seeing if the topic models built are robust enough to match a thread with an e-mail that has *not* been seen. Further, in the interest of generalizing results to other conversational domains such as chat, blog comments, or phone conversations, they do not in general have the luxury of quoted reply text.

For our baseline, and for creation of the training and test data set, the atomic unit of analysis

was the message. Messages vary greatly in their lengths, and our results did not account for this factor.

Based on these observations, we improve thread classification by building better data sets, studying multiple individuals' e-mail corpora, and better parametrizing their nature and variance. If there is enough commonality in the topics discussed by individuals within certain demographics, training on multiple peoples' e-mail would help to refine the topic models and improve classification capability.

5.1.1 Distance Metrics

For the topic variation experiment with 1000 processed threads, the Shannon and Cosine measures diverged notably as the number of topics grew, performing consistently better as the number of topics increased. As well, for the 100 processed threads, Shannon and Cosine measures performed better than their counterparts.

The Shannon and Cosine measures also perform better than Euclidean distance and KL divergence for the thread variation experiment with processed threads. For the raw threads, there is no clear dominant performer.

The 1000 thread experiment with the raw threads shows the Shannon and KL divergence outperformed Cosine and Euclidean measures consistently for all topics, while the results for 100 raw threads are too close for a winner to be declared.

5.2 Topic Change

While the keyword clustering experiment generally had poor results, many threads did have clusters that resulted in topics with clear boundaries, and contained close to if not precisely the correct number of topic change events.

The range of F-score results for the different experiments was from 0 to 1. Most of the threads where all changes were accurately detected were those with a single change, and strong topics. Still, while there were a non-negligible number of finite results, the more *frequent* result was 0, meaning that 0 topic changes were accurately detected in the correct location. Here we explore some of the factors in the success and failure of the techniques applied.

The runs with $N = 25$ performed better overall, although changing N from 25 to 10 had a smaller effect on the F-scores than varying the DBSCAN inputs. In both cases the, ($\epsilon = 6$,

$min\_pts = 4$) values performed better than ($\epsilon = 10$, $min\_pts = 6$). Although $\epsilon = 10$ is a more forgiving radius, the topic data was too sparse for $min\_pts = 6$. For these experiments, often no clusters were even detected.

Our corpus was relatively small, with a large breadth of topic coverage and short, terse e-mails on average. Although there were a few overarching themes across the corpus resulting in good models for a few topics, many of the other topics were poorly reflected in the topic models, with keywords not accurately represented. Correspondingly, words in the message bodies that perhaps should have been associated with a topic were missed, and the resulting clusters were of smaller size. In other cases, words had not developed a clear association with any topic, and topics that were seen to exist by human inspection were consistently treated as noise by the algorithms. Certain threads never had good cluster representation.

This result could simultaneously be indicative of characteristic topic lengths in this medium. Rather than being a descriptive prose environment, most e-mail tends to be purposeful and functional (there are, of course, exceptions). Even a 7- or 8-message thread might only have a couple of thousand characters, leaving little time for a topic to develop.

The (25, None, 5, 3) experiment increased the number of keywords again, removed the keyword threshold altogether, and then shrank the DBSCAN radius. Relaxing the threshold allowed more of the significant keywords to be included, in order to see if the topic boundaries would be more accurate. What we saw is that slightly more threads seemed to have non-zero results in this case, although they were on average lower than the non-zero results of the other runs. It seemed that the parameters for this run were a little too forgiving, often including words it shouldn't have.

There were several threads which consistently had high F-scores across multiple different parameter inputs. These topics may have been better represented in the training corpus, and therefore while different parameters shifted the topic boundaries forward or back a few words, or changed the density of keywords in the cluster, the clusters themselves persisted. Overall, these more frequent topics were robust under different sets of parameter inputs, and performed well in these experiments.

For some threads, the number of topic changes detected was correct, but the location was incorrect because the clusters were too far apart. In these cases, the 'median' method used to identify transitions points (whereby the middle point between two cluster boundaries was taken) was at

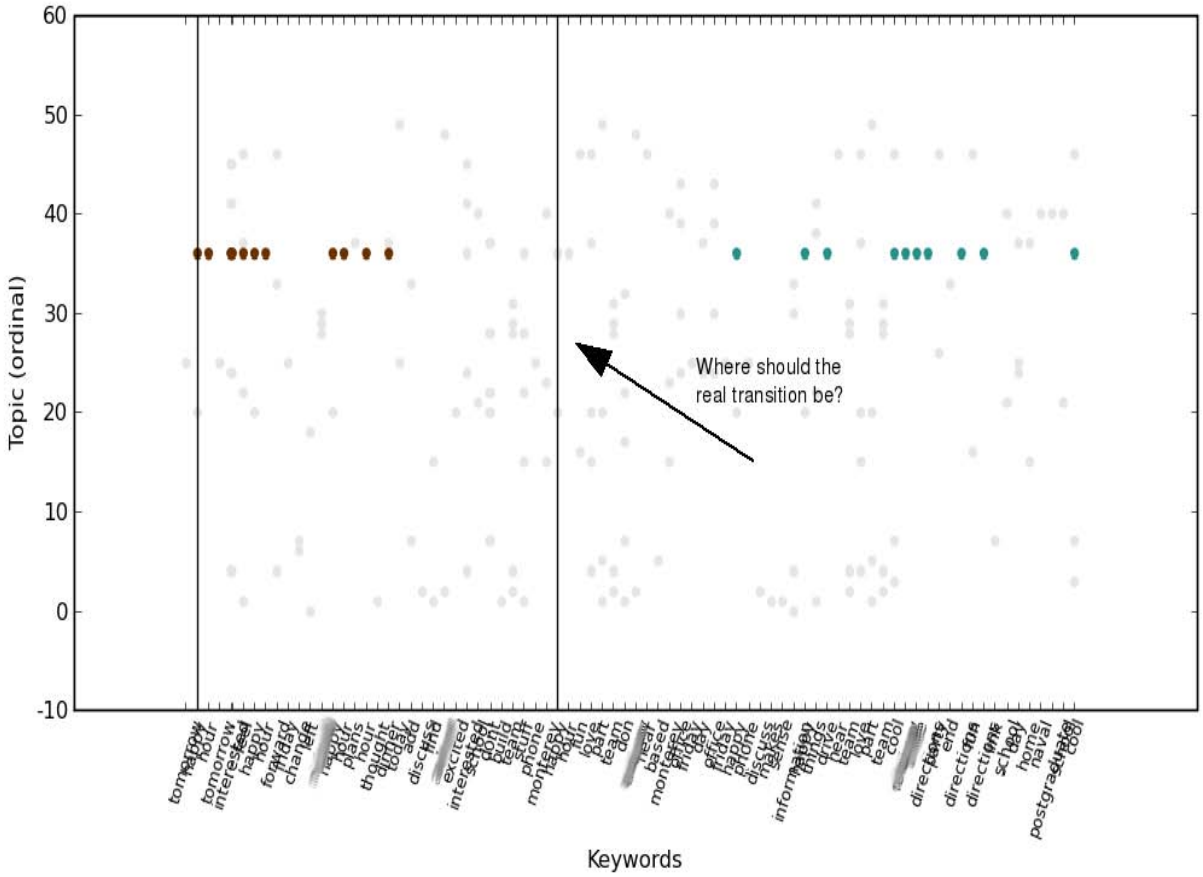

Figure 5.1: DBSCAN clusters are too far apart, resulting in meaningless transition.

fault, causing transitions to be placed far from the boundaries of either cluster. See Figure 5.1 for an example of this. These smaller clusters were often more accurate, but their boundaries simply too tight. The resulting space between clusters introduced ambiguity about where the transition occured. In general, the location and size of neighbouring clusters greatly affected the transition locations.

In other cases, a cluster ended, and no subsequent cluster was identified, but the transition to 'noise' after the end of the last cluster *was* a meaningful transition from one topic that was rather well defined, to one that simply was not. This could be remedied by either or both of more suitable clustering parameters, as well as more nuanced transition point identification (see Figure 5.2).

Certain transitions were incorrectly inserted between clusters of the same topic. In these cases the division between clusters was an artifact of the parameters for that particular run of the experiment, and did not represent the presence of intermediary topics. See again Figure 5.2

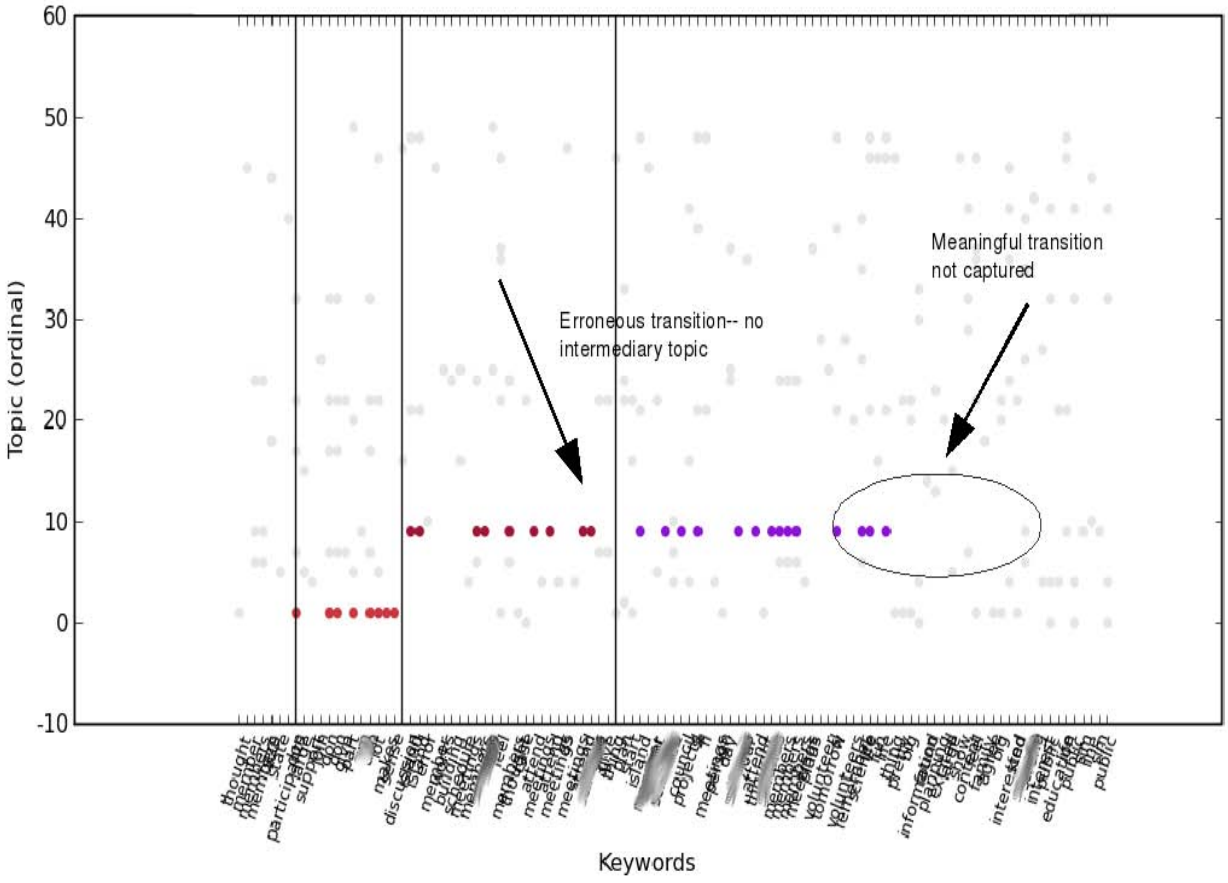

Figure 5.2: The end of the last cluster is a meaningful transition, but is not captured because there is no subsequent cluster. The transition identified *between* clusters of the same topic is an artifact of the experiment parameters, and not a real transition.

The transition mechanism also did not perform well with overlapping clusters. In the present implementation, multiple clusters overlapping would cause the endpoints of those clusters to be averaged, when calculating the transition point between the end of one set of clusters and the beginning of another. This made the calculation simple, but there was no particular structural reason for these clusters to be handled that way. A more subtle treatment of transition point calculation would aid in the identification of accurate transitions.

Beyond the mechanics of topic change identification, there are subtleties to its definition as well. Almost never is a transition discrete, but rather it is more accurately modelled by the decreasing influence of one topic, and the corresponding increasing influence of another. An intuitive decision for the transition point in this case would be the point at which one topic overtakes another in terms of weight. But clearer steps should be taken to make logical decisions when there is a gap between clusters, or when the topic transion is gradual.

For both experiments, the transition point selection mechanism involved taking the median point either of the sliding window, or the boundaries of two clusters. For the sliding window approach, this is clearly not a scalable approach if the window size were to get larger. In fact, this is a shortfall of the method overall, since by taking the total weight over a single window, the growing influence of secondary topics is smoothed over, as is the order of the words (for example, perhaps the winning topic is actually strong at the beginning of the window but tapers off at the end). A way to address this might be to keep track of multiple topics in each window, supporting the conceptual approach that documents are mixtures of topics.

This thesis treated transition points between topics as discrete points in time. A more nuanced analysis of how topic changes, alternative ways to measure it, and ways to represent slow versus fast changes, or discrete versus evolutionary changes, would enrich the discussion.

Overall, the sliding window technique performed measureably worse, in particular in the number of changes detected (as opposed to their location). For many of the threads, this method vastly ovestimated the number of transitions (see Figure 5.3 C), causing overall high recall. The result in these cases was that the likelihood of one of those transitions being in the correct location increased, but the positive results in those cases were actually a side effect of the recall.

Unfortunately, increasing the window size from 20 to 40 and then to 100 did more to negate previously correct transitions detected by the smaller window sizes, than to improve the vast overfitting observed in some threads.

Between the keyword clustering and sliding window approaches, the sliding window approach was simpler to implement, but too sensitive to change. The keyword clustering approach was more challenging, with more parameters to understand and refine, but it was also better at detecting substance over noise. The DBSCAN method did not account for word-topic weights beyond their presence in the top N keywords. There was a question of whether this would gloss over important nuances, but it did not negatively impact the results. In fact, because the top

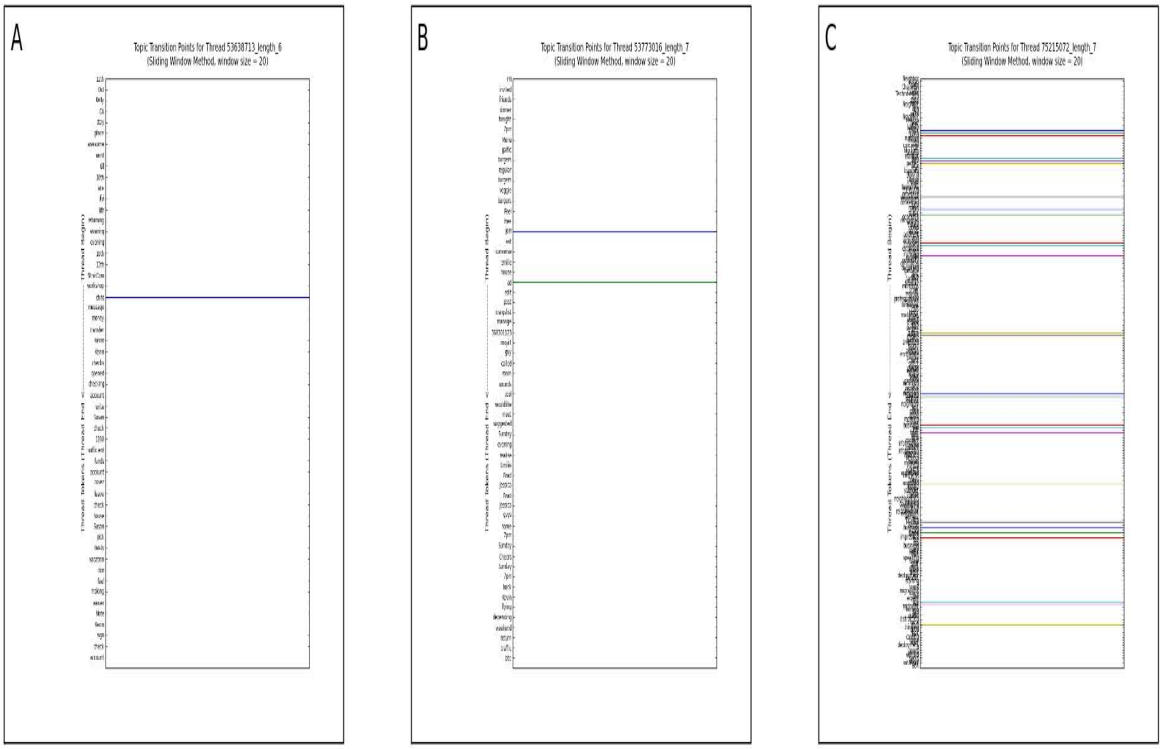

Figure 5.3: Three outputs from the sliding window experiment. Window sizes as described in respective titles. A) Correct results; B) Erroneously detects two topics, close to the actual topic change; C) Extreme overfitting, too many topics detected. LDA Topics = 50). Personally identifying terms are grayed out for privacy purposes.

words for a topic tend to have extremely high weights (again, see Section 4.4) it raises the question of whether the sliding window approach is *too* influenced by the relative weights of words, especially as they get large. Future experiments might try modifying the sliding window approach by capping or smoothing the word-topic weights.

Topic change is a more challenging area than e-mail-thread classification. The classification task had the advantage of working with all information in a thread, whereas the topic detection was working with shorter sequences of words; sometimes only a couple of sentences. This isn't much content for the algorithm to accurately detect a topic, especially if the words used were not already heavily weighted towards a particular topic. Oftentimes, it might be missed altogether.

To a certain extent, using e-mail conflated many of the questions about how to detect topic change. Most messages are short, and topic changes, when they do occur, can be very subtle. Between the data cleaning and pre-processing issues discussed in the previous section, the multiple possible paths through each thread, and the low frequency with which clear topic change occurs in e-mail, it was difficult to get good quality data, in enough quantity, to make statistical conlusions about the result.

In summary, detecting the presence of topic change in a conversational setting still requires work, as evidenced by the results and discussions above. Many of the observations made here could likely improve the techniques, but first and foremost the current techniques should be applied to a larger corpus, and run with a significantly expanded range of parameters inputs. Additionally, a more formal analysis of how to optimize the paramters N, T, ϵ and min_pts used in the keyword clustering experiment is necessary.

5.2.1 Open Questions

Through this experiment, many important questions surfaced about the nature of topic change and how it should be measured. Every topic is arguably part of a hierarchy, and every topic has sub-elements. As a result, a critical factor in topic change analysis is the coarseness of analysis applied, and defining transitions that are formalizable and repeatable.

LDA is a tool we use to statistically represent topics, and to some degree the number of topics selected as input parameter to the algorithm will determine this coarseness. At the same time, it should be remembered that the 'topics' discovered by LDA might not be the topics a human reader would pick out.

For example, an entire e-mail thread might be about an academic course being taken, but the first few e-mails are about meeting for a study session, and the last few about the latest assignment. Similarly, another thread might be about a vendor's plans to film an event, but the beginning might be about getting the vendor event passes, and the second half might be a detailed discussion about the cabling they need. Less explicit changes need more training.

Additionally, there are many topics which appear once in the corpus and then never again, and thus as mentioned above, the weighting of those words for the respective topics isn't very good. It is easy to say that more and better data would help in our ability to detect more subtle transitions, and that is true. But it would also be very interesting to take multiple conversational corpora and train over the whole set, in an attempt to improve the quality of the models. If there

genuinely is overlap in conversational topics across individuals, this will be helpful. And where there is not, such a technique might instead be useful in identifying personal versus popular topics.

Proximity between words of the same topic could be used to scale the weight of each word in either technique, decreasing the likelihood of stray words being picked up. In fact, the inherent time dimension of conversational texts suggests that a Markov approach giving greater weight to the current or previously seen topics, could help to improve the results. Topics could be clustered together, showing where transitions among certain sets of topics are more likely than others. The likelihoods would ultimately be a function of generic as well as personal categories.

Part-of-speech tagging could be used to give grammatical context where token context is insuffient. In particular, conversational contexts contain many question/answer type interactions, where even the probabilistic models applied here seem to fail. Modeling these interactive dynamics and using them as, or to inform, the priors for topics might yield improved results.

5.3 Stopwords and Keywords

Additional work on averaging and entropy-based methods (such as [21]) could determine and remove stop words on the fly. This would reflect the fact that stop words are often a function of context, and would also support the development of language independent solutions.

Consider Figure 5.4, which shows a full graph of LDA word-topic weights, where stop words have not been removed. Although the stop words are co-appearing with their more substantive neighbours, they are also co-appearing frequently with other stopwords. In the resulting LDA topic models, these stop words are weighted most heavily towards topics which appear to be *dominated* by stop words. This could be used as a way to programmatically identify stop words.

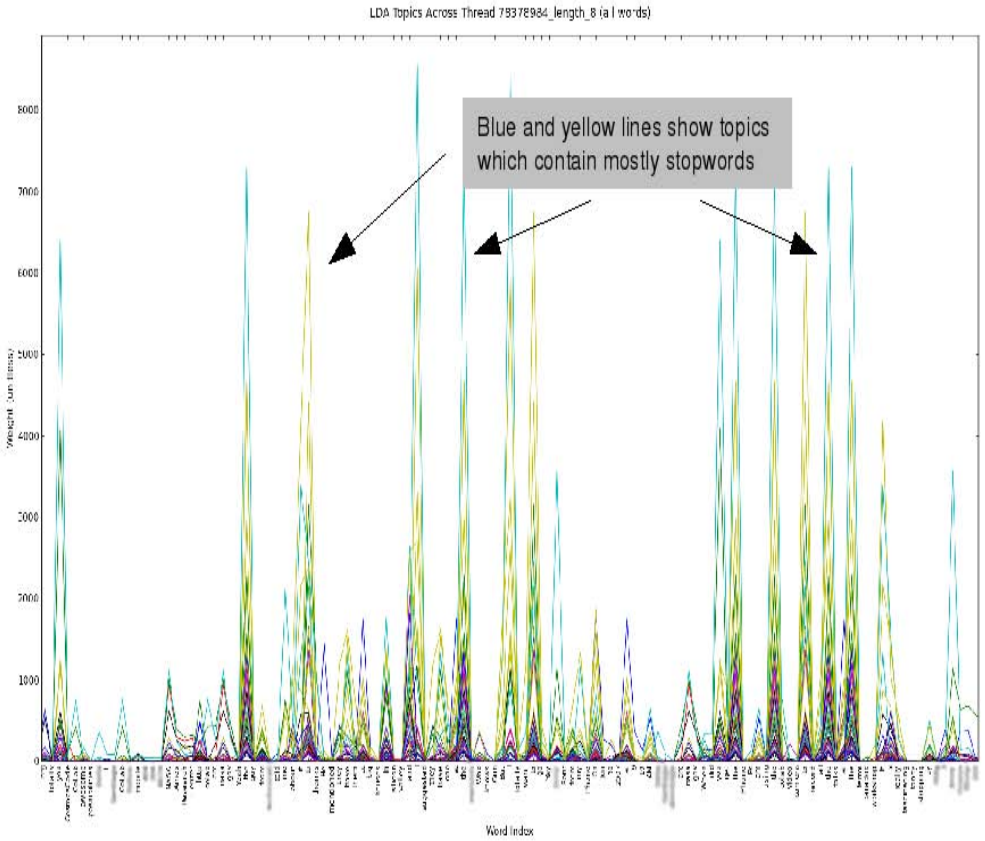

Figure 5.4: In this graph of a subsection of a thread, we can see that stopwords consistently appear in two topics above and beyond all others. Personally identifying names and locations have been blurred.

5.4 Signature Detection

Signature detection and removal is a laborious and time consuming process if done by hand, and error-prone if done programmatically. In the course of examining patterns present in e-mail threads, small repeated blocks of text, in particular, signatures, were seen to be very distinct when viewed visually. In addition, because signatures of frequent correspondents appear multiple times, in exactly the same word sequences, these signatures are clustered together in topics with extremely high frequency (see 5.6).

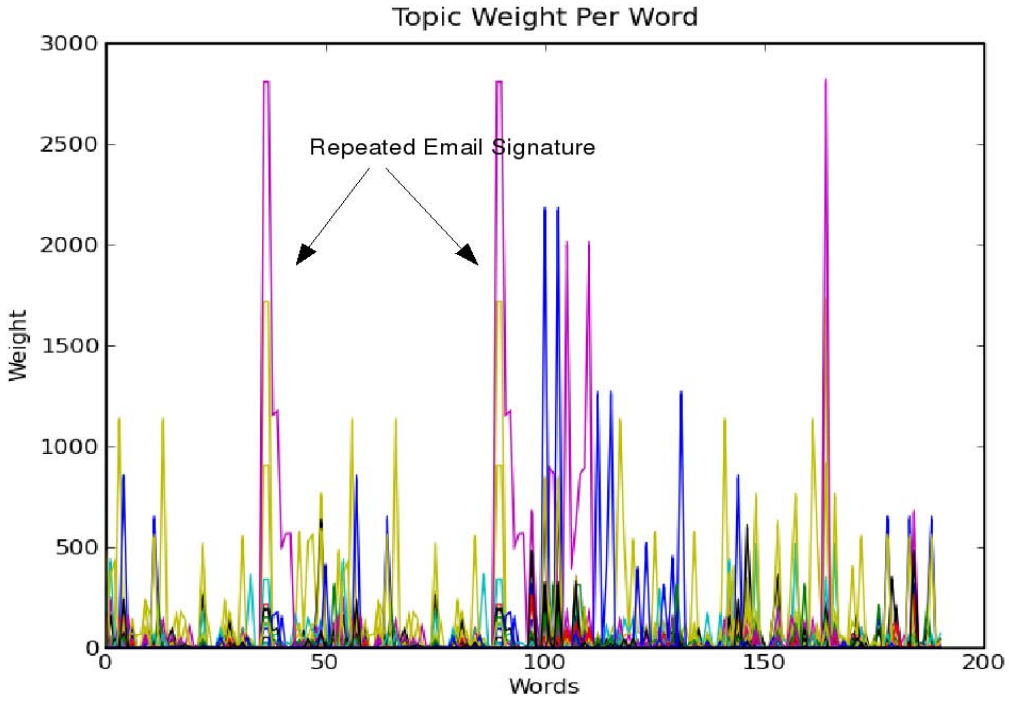

Figure 5.5: A graph of topic word weight for one e-mail. Again the words along the x-axis are in order of appearance, so this dimension also represents time.

This begs the question of whether LDA might be useful in automatically identifying and removing signatures. This would only be useful for authors with whom correspondence was moderately frequent. However, others have shown [18] that the relationship between correspondents and frequency exhibits an exponential falloff, suggesting that such a method, if it succeeded, would still be useful for the majority of e-mails in an individual's corpus.

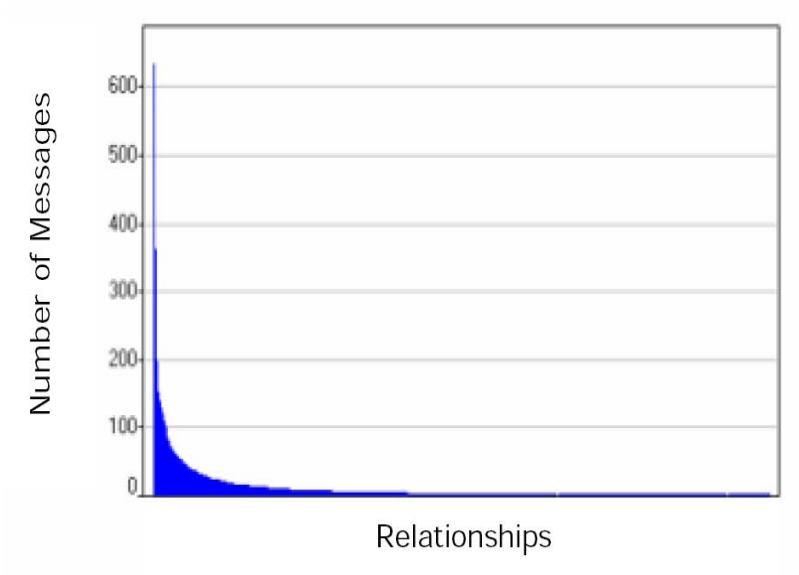

Figure 5.6: Power distribution of relationships in e-mail. Figure from [18, p.8]

5.5 Other Future Work

Many times in this document the nature of conversational text compared to more traditional documents has been mentioned. E-Mail has more stop words, more implicit references, and is more casual. It is possible these same characteristics which make e-mail challenging to analyze would also allow conversational text to be automatically identified in a stream of non-conversational text. For example, could the point in a webpage where a blog posts ends and the comments begin, be automatically identified? Or conversational content in noisy packet streams? Similarly, while conducting forensic file system analysis or file carving, such a technique could identify chat logs or e-mails not otherwise known to exist.

While exploring the optimal number of input topics to the LDA algorithm, a question that arises is whether there might be a characteristic number of topics which map onto the average individual's day-to-day communications. Do most people talk about a certain number of things? What can be said of those who have a larger or smaller breadth of topics they discuss? Might conditions such as Autism or Attention Deficit Disorder be detected via long term studies of topics in individuals' personal communications?

CHAPTER 6:
Conclusions

We have demonstrated that state-of-the-art probabilistic topic models can be successfully applied to classifying emails with their original threads. These documents are more casual and contextual, and generally shorter and more terse, than their more formal counterparts. For up to 1000 threads, we show that raw email messages can be correctly classified with up to 95% accuracy.

For more generic conversational threads, without the characteristic quoted and reply text of emails, we present results significantly better than baseline, and identify several ways that the results could be improved upon.

Further exploring the nature of conversation corpora, two new techniques for identifying topic change are developed and tested on a token set of cleaned email threads. The first is a keyword clustering method, and the second is a sliding window technique. The results show promise, and provide a concrete baseline on which future work can improve. We describe numerous ways in which these methods could be refined.

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

[1] Natural language toolkit, 2009. http://www.nltk.org.

[2] D.M. Blei and J.D. Lafferty. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pp. 113–120. ACM New York, NY, USA, 2006.

[3] D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

[4] G. Cselle, K. Albrecht, and R. Wattenhofer. BuzzTrack: topic detection and tracking in email. In *Proceedings of the 12th international conference on Intelligent user interfaces*, pp. 190–197. ACM New York, NY, USA, 2007.

[5] M. Damashek. Gauging similarity with n-grams: Language-independent categorization of text. *Science*, 267(5199):843–848, 1995.

[6] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Unknown*.

[7] M. Ester, H.P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining, Portland, OR, AAAI Press*, pp. 226–231. AAAI Press, 1996.

[8] NASA Headquarters Garth Henning. Personal communication.

[9] T. Griffiths and M. Steyvers. A probabilistic approach to semantic representation. In *Proceedings of the 24th annual conference of the cognitive science society*, pp. 381–386. Citeseer, 2002.

[10] T. Griffiths and M. Steyvers. Prediction and semantic association. *Advances in neural information processing systems*, pp. 11–18, 2003.

[11] T.L. Griffiths and M. Steyvers. Finding scientific topics, 2004, National Acad Sciences.

[12] S. Havre, B. Hetzler, and L. Nowell. ThemeRiver: Visualizing theme changes over time. In *IEEE Symposium on Information Visualization, 2000. InfoVis 2000*, pp. 115–123, 2000.

[13] M.A. Hearst. TextTiling: Segmenting text into multi-paragraph subtopic passages. *Computational linguistics*, 23(1):33–64, 1997.

[14] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 50–57. ACM New York, NY, USA, 1999.

[15] B. Klimt and Y. Yang. Introducing the Enron corpus. In *First conference on email and anti-spam (CEAS)*, 2004.

[16] Craig Martell. *FORM: An Experiment in the Annotation of the Kinematics of Gesture*. PhD thesis, University of Pennsylvania, 2005.

[17] Andrew Kachites McCallum. Mallet: A machine learning for language toolkit, 2002. http://mallet.cs.umass.edu.

[18] A. Perer, B. Shneiderman, and D.W. Oard. Using rhythms of relationships to understand e-mail archives. *Journal of the American Society for Information Science and Technology*, 57(14), 2006.

[19] M. Steyvers and T. Griffiths. Probabilistic topic models. *Handbook of Latent Semantic Analysis*, p. 427, 2007.

[20] M. Steyvers, P. Smyth, M. Rosen-Zvi, and T. Griffiths. Probabilistic author-topic models for information discovery. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 306–315. ACM New York, NY, USA, 2004.

[21] Jenny Tam. Detecting age in online chat. Master's thesis, Naval Postgraduate School, Monterey, CA, 2009.

[22] F.B. Viégas, M. Wattenberg, and K. Dave. Studying cooperation and conflict between authors with history flow visualizations. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 575–582. ACM New York, NY, USA, 2004.

APPENDIX A:
Code

A.1 DBSCAN

The DBSCAN Python implementation used for this thesis is included below.

```
#!/usr/bin/python

##################################################
# Jessy Cowan-Sharp, August 2009
# References:
#   1. "A density-based algorithm for discovering clusters in large
#      spatial databases with noise," Ester, M. and Kriegel, H.P. and
#      Sander, J. and Xu, X.

##################################################

# very code-like pseudo-code
# DBSCAN
# for point in points:
#    if point is visited:
#        continue
#    mark point as visited
#    neighbours = immediate_neighbours(point, epsilon)
#    if len(neighbours) > min_pts:
#        cluster = new_cluster()
#        append point to cluster
#        for n in neighbours:
#            cluster.append(all_neighbours(n))
#    else:
#        mark point as NOISE
#
# def all_neighbours(n, epsilon, cluster):
#     for point in points:
#         if point has not been visited:
#             mark point as visited
#             new_points = immediate_neighbours(point)
#             if len(new_points) > min_pts:
#                 points.append(new_points)
#         if point is not member of any cluster:
#             append point to cluster

from math import pow, sqrt

class Point(object):
    ''' internal helper class to support algorithm implementation'''
    def __init__(self,feature_vector):
        # feature vector should be something like a list or a numpy
        # array
        self.feature_vector = feature_vector
        self.cluster = None
        self.visited = False

    def __str__(self):
        return str(self.feature_vector)

def _as_points(points):
    ''' convert a list of list- or array-type objects to internal
    Point class'''
    return [Point(point) for point in points]
```

```
def as_lists(clusters):
    ''' converts the Points in each cluster back into regular feature
    vectors (lists).'''
    clusters_as_points = {}
    for cluster, members in clusters.iteritems():
        clusters_as_points[cluster] = [member.feature_vector for member in members]
    return clusters_as_points

def print_points(points):
    ''' klugey function for printing lists of points. '''
    s = ''
    for p in points:
        s += str(p) + '\n'
    return s[:-2]

def euclidean(x,y):
    ''' calculate the euclidean distance between x and y.'''
    # sqrt((x0-y0)^2 + ... (xN-yN)^2)
    assert len(x) == len(y)
    sum = 0.0
    for i in xrange(len(x)):
        sum += pow(x[i] - y[i],2)
    return sqrt(sum)

def immediate_neighbours(point, all_points, epsilon, distance, debug):
    ''' find the immediate neighbours of point.'''
    # NOTE: there is probably a better way to do this.
    neighbours = []
    for p in all_points:
        if p == point:
            # you cant be your own neighbour...!
            continue
        d = distance(point.feature_vector,p.feature_vector)
        if d < epsilon:
            neighbours.append(p)
    return neighbours

def add_connected(points, all_points, epsilon, min_pts, current_cluster, distance, debug):
    ''' find every point in the set of all_points which are
    density-connected, starting with the initial points list. '''
    cluster_points = []
    for point in points:
        if not point.visited:
            point.visited = True
            new_points = immediate_neighbours(point, all_points, epsilon, distance, debug)
            if len(new_points) >= min_pts:
                # append any new points on the end of the list we're
                # already iterating over.
                for p in new_points:
                    if p not in points:
                        points.append(p)

        # here, we separate 'visited' from cluster membership, since
        # 'visited' only helps keep track of if we've checked this
        # point for neighbours. it may or may not have been assessed
        # for cluster membership at that point.
        if not point.cluster:
            cluster_points.append(point)
            point.cluster = current_cluster
    if debug:
        print 'Added points %s' % print_points(cluster_points)
    return cluster_points

def dbscan(points, epsilon, min_pts, distance=euclidean, debug=False):
    ''' Main dbscan algorithm function. pass in a list of feature
    vectors (most likely a list of lists or a list of arrays), a
    radius epsilon within which to search for neighbouring points, and
    a min_pts, the minimum number of neighbours a point must have
    within the radius epsilon to be considered connected. the default
```

```
    distance metric is euclidean, but another could be used as
    well. your custom distance metric must accept two equal-length
    feature vectors as input as return a distance value. pass in
    debug=True for verbose output.'''

    assert isinstance(points, list)
    epsilon = float(epsilon)
    if not isinstance(points[0], Point):
        # only check the first list instance. imperfect, but the lists
        # could be arbitrarily long.
        points = _as_points(points)

    if debug:
        print '\nEpsilon: %.2f' % epsilon
        print 'Min_Pts: %d' % min_pts

    clusters = {}     # each cluster is a list of points
    clusters[-1] = [] # store all the points deemed noise here.
    current_cluster = -1
    for point in points:
        if not point.visited:
            point.visited = True
            neighbours = immediate_neighbours(point, points, epsilon, distance, debug)
            if len(neighbours) >= min_pts:
                current_cluster += 1
                if debug:
                    print '\nCreating new cluster %d' % (current_cluster)
                    print '%s' % str(point)
                point.cluster = current_cluster
                cluster = [point,]
                cluster.extend(add_connected(neighbours, points, epsilon, min_pts,
                                             current_cluster, distance, debug))
                clusters[current_cluster] = cluster
            else:
                clusters[-1].append(point)
                if debug:
                    print '\nPoint %s has no density-connected neighbours.' % str(point.feature_vector)

    # return the dictionary of clusters, converting the Point objects
    # in the clusters back to regular lists
    print 'length of original list: %d' % len(points)
    returned = 0
    for members in clusters.values():
        returned += len(members)
    print 'length of returned points: %d' % returned
    return as_lists(clusters)

if __name__ == '__main__':

    import random

    epsilon = 2.0
    min_pts = 2.0
    points = []
    points.append([1,1])
    points.append([1.5,1])
    points.append([1.8,1.5])
    points.append([2.1,1])
    points.append([3.1,2])
    points.append([4.1,2])
    points.append([5.1,2])
    points.append([10,10])
    points.append([11,10.5])
    points.append([9.5,11])
    points.append([9.9,11.4])
    points.append([15.0, 17.0])
    points.append([15.0, 17.0])
    points.append([7.5, -5.0])

    clusters = dbscan(points, epsilon, min_pts, debug=True)
```

```
print '\n========== Results of Clustering ============'
for cluster, members in clusters.iteritems():
    print '\n--------Cluster %d---------' % cluster
    for point in members:
        print point

points = []
for i in xrange(100):
    points.append([random.uniform(0.0, 20.0), random.uniform(0.0, 20.0)])

clusters = dbscan(points, epsilon, min_pts, debug=True)
print '\n========== Results of Clustering ============'
for cluster, members in clusters.iteritems():
    print '\n--------Cluster %d---------' % cluster
    for point in members:
        print point
```

APPENDIX B:
Stop Words

B.1 Generic Stop Words

The generic English language stop word list was taken from the Natural Language Toolkit (NLTK) [1]. It is comprised of the following words:

a	amongst	away	c	could
a’s	an	awfully	c’mon	couldn’t
able	and	b	c’s	course
about	another	be	came	currently
above	any	became	can	d
according	anybody	because	can’t	definitely
accordingly	anyhow	become	cannot	described
across	anyone	becomes	cant	despite
actually	anything	becoming	cause	did
after	anyway	been	causes	didn’t
afterwards	anyways	before	certain	different
again	anywhere	beforehand	certainly	do
against	apart	behind	changes	does
ain’t	appear	being	clearly	doesn’t
all	appreciate	believe	co	doing
allow	appropriate	below	com	don’t
allows	are	beside	come	done
almost	aren’t	besides	comes	down
alone	around	best	concerning	downwards
along	as	better	consequently	during
already	aside	between	consider	e
also	ask	beyond	considering	each
although	asking	both	contain	edu
always	associated	brief	containing	eg
am	at	but	contains	eight
among	available	by	corresponding	either

else
elsewhere
enough
entirely
especially
et
etc
even
ever
every
everybody
everyone
everything
everywhere
ex
exactly
example
except
f
far
few
fifth
first
five
followed
following
follows
for
former
formerly
forth
four
from
further
furthermore
g
get
gets
getting
given
gives
go
goes
going
gone
got
gotten
greetings
h
had
hadn’t
happens
hardly
has
hasn’t
have
haven’t
having
he
he’s
hello
help
hence
her
here
here’s
hereafter
hereby
herein
hereupon
hers
herself
hi
him
himself
his
hither
hopefully
how
howbeit
however
i
i’d
i’ll
i’m
i’ve
ie
if
ignored
immediate
in
inasmuch
inc
indeed
indicate
indicated
indicates
inner
insofar
instead
into
inward
is
isn’t
it
it’d
it’ll
it’s
its
itself
j
just
k
keep
keeps
kept
know
knows
known
l
last
lately
later
latter
latterly
least
less
lest
let
let’s
like
liked
likely
little
look
looking
looks
ltd
m
mainly
many
may
maybe
me
mean
meanwhile
merely
might
more
moreover
most
mostly
much
must
my
myself
n
name
namely
nd
near
nearly
necessary
need
needs
neither
never
nevertheless
new
next

nine
no
nobody
non
none
noone
nor
normally
not
nothing
novel
now
nowhere
o
obviously
of
off
often
oh
ok
okay
old
on
once
one
ones
only
onto
or
other
others
otherwise
ought
our
ours
ourselves
out
outside
over
overall
own
p
particular
particularly
per
perhaps
placed
please
plus
possible
presumably
probably
provides
q
que
quite
qv
r
rather
rd
re
really
reasonably
regarding
regardless
regards
relatively
respectively
right
s
said
same
saw
say
saying
says
second
secondly
see
seeing
seem
seemed
seeming
seems
seen
self
selves
sensible
sent
serious
seriously
seven
several
shall
she
should
shouldn’t
since
six
so
some
somebody
somehow
someone
something
sometime
sometimes
somewhat
somewhere
soon
sorry
specified
specify
specifying
still
sub
such
sup
sure
t
t’s
take
taken
tell
tends
th
than
thank
thanks
thanx
that
that’s
thats
the
their
theirs
them
themselves
then
thence
there
there’s
thereafter
thereby
therefore
therein
theres
thereupon
these
they
they’d
they’ll
they’re
they’ve
think
third
this
thorough
thoroughly
those
though
three
through
throughout
thru
thus
to
together
too
took

toward	used	we'll	wherever	would
towards	useful	we're	whether	wouldn't
tried	uses	we've	which	x
tries	using	welcome	while	y
truly	usually	well	whither	yes
try	uucp	went	who	yet
trying	v	were	who's	you
twice	value	weren't	whoever	you'd
two	various	what	whole	you'll
u	very	what's	whom	you're
un	via	whatever	whose	you've
under	viz	when	why	your
unfortunately	vs	whence	will	yours
unless	w	whenever	willing	yourself
unlikely	want	where	wish	yourselves
until	wants	where's	with	z
unto	was	whereafter	within	zero
up	wasn't	whereas	without	
upon	way	whereby	won't	
us	we	wherein	wonder	
use	we'd	whereupon	would	

B.2 Custom Stop Words

These stop words were identified by human inspection of the corpus.

jessy	www	gmail	4	13
cowan	org	google	5	14
sharp	com	googlegroups	6	15
cowansharp	net	groups	7	16
cowan-sharp	arc	subscribe	8	17
202	nasa	unsubscribe	9	18
360	gov	1	10	19
3967	mail	2	11	20
http	email	3	121	80

30	650	mailman	br
2006	815	listinfo	ll
2007	450	lists	ve
2008	831	cgi	div
2009	656	bin	ames
html	mailing	604	center

THIS PAGE INTENTIONALLY LEFT BLANK

Referenced Authors

Albrecht, K. 8, 11

Blei, D.M. 4, 7–9, 16

Cselle, G. 8, 11

Damashek, M. 8, 28
Dave, K. 10
Deerwester, S. 8
Dumais, S.T. 8

Ester, M. ix, 20–22

Furnas, G.W. 8

Garth Henning,
NASA Headquarters 4
Griffiths, T. ix, 16–18, 30
Griffiths, T.L. 17

Harshman, R. 8
Havre, S. 10
Hearst, M.A. 3, 7, 8
Hetzler, B. 10
Hofmann, T. 8

Jordan, M.I. 8, 16

Klimt, B. 11
Kriegel, H.P. ix, 20–22

Lafferty, J.D. 4, 7, 9
Landauer, T.K. 8

Martell, Craig 23
McCallum, Andrew Kachites 29

Ng, A.Y. 8, 16
Nowell, L. 10

Oard, D.W. x, 60, 61

Perer, A. x, 60, 61

Rosen-Zvi, M. 18

Sander, J. ix, 20–22
Shneiderman, B. x, 60, 61
Smyth, P. 18
Steyvers, M. ix, 16–18, 30

Tam, Jenny 3, 37, 58

Viégas, F.B. 10

Wattenberg, M. 10
Wattenhofer, R. 8, 11

Xu, X. ix, 20–22

Yang, Y. 11

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Elver, Virginia
2. Loudly Knox Library
   Naval Postgraduate School
   Monterey, California
3. Marine Corps Representative
   Naval Postgraduate School
   Monterey, California
4. Directory, Training and Education, MCCDC, Code C46
   Quantico, Virginia
5. Marine Corps Tactical System Support Activity (Attn: Operations Officer)
   Camp Pendleton, California